Summer 8-13-2020

Routing Optimization in Heterogeneous Wireless Networks for Space and Mission-Driven Internet of Things (IoT) Environments

Sara El Alaoui
University of Nebraska-Lincoln, ea.sara@huskers.unl.edu

ROUTING OPTIMIZATION IN HETEROGENEOUS WIRELESS NETWORKS FOR SPACE AND MISSION-DRIVEN INTERNET OF THINGS (IoT) ENVIRONMENTS

by

Sara El Alaoui

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Engineering

(Computer Engineering–Computer Science)

Under the Supervision of Professor Byrav Ramamurthy

Lincoln, Nebraska

July, 2020

ROUTING OPTIMIZATION IN HETEROGENEOUS WIRELESS NETWORKS FOR SPACE AND MISSION-DRIVEN INTERNET OF THINGS (IoT) ENVIRONMENTS

Sara El Alaoui, Ph.D.

University of Nebraska, 2020

Adviser: Byrav Ramamurthy

As technological advances have made it possible to build cheap devices with more processing power and storage, and that are capable of continuously generating large amounts of data, the network has to undergo significant changes as well. The rising number of vendors and variety in platforms and wireless communication technologies have introduced **heterogeneity** to networks compromising the efficiency of existing routing algorithms. Furthermore, most of the existing solutions assume and require connection to the backbone network and involve changes to the infrastructures, which are not always possible – a 2018 report by the Federal Communications Commission shows that over 31% of the population living in rural areas has little to no broadband coverage.

In this dissertation, we study routing optimization in heterogeneous wireless networks in order to fill this gap in research and properly address the challenges they pose. We first propose a **novel mathematical classification** based on their **contacts** (i.e. communication windows between two network entities) in order to aid routing. We define four types of contacts: predicted, scheduled, discovered and continuous. Next, we investigate single-attribute and multi-attribute message scheduling and routing in scheduled contacts using Space Networks as a case study and proposed N-Look Ahead Routing and scheduling Algorithm (**N-**

LARS) and the Multi-Attribute Routing Algorithm (**MARS**). We then study all four contact types and develop a statistical analysis framework (**STAN**), and predictive routing algorithm, **PETRA**. We evaluated our work on a disaster recovery Mission-Driven IoT (MD-IoT) network. Finally, we consider predicted and continuous contacts by designing and formulating the **low-latency routing problem as QIP and ILP models** and by developing an edge computing framework, **ERGO**, which we applied on Agricultural-IoT networks.

Through this dissertation on routing in heterogeneous wireless networks for space and Mission-Driven IoT, we show that precise modeling of network heterogeneity properties enables us to enhance network performance in terms of various metrics. By using different tools including machine learning, edge computing, statistical analysis, MADM and age of information, we demonstrate that heterogeneity and the lack of network infrastructure can be overcome paving the way for heterogeneous wireless networks that are highly efficient and dynamic.

ACKNOWLEDGMENTS

I would like to express my gratitude and sincere thanks to my advisor Dr. Byrav Ramamurthy for his support and help. His patience and kindness have taught me how to aspire for success while enjoying every step on the path leading to it. His guidance has been essential in the completion of this work.

I would also like to acknowledge Mr. Deepak Nadig (UNL), Dr. Santoch Pitla (UNL), Mr. Scott Burleigh and Mr. Philip Tsao (NASA Jet Propulsion Lab, in Pasadena California) for their contributions and thank them for the inspirational discussions and their feedback about our work. This work was supported in part by NSF CNS-1345277 and by a research mini-grant from NASA Nebraska EPSCOR. Mr. Deepak Nadig, who developed the ERGO edge computing platform (see Chapter 5) and Dr. Santosh Pitla are co-authors of a pending paper on Ag-IoT.

I want to thank the committee members, Dr. Massimiliano Pierobon, Dr. Lisong Xu and Dr. Wei Qiao, for their time and comments that helped enhance the quality of this work.

I extend my thanks to Dr. Naeem Sheikh for him precious help and to all my friends and family; namely, Dr. Zahmeeth Sakkaff, Ms. Seynabou, Ms. Salma, Ms. Hind, Ms. Mariana and Dr. Adrian Lara, Ms. Soumaya, all Netgroup members I had the pleasure to work with, and great friends I met at UNL.

This journey would not have been possible without the love, support and sacrifices of my beloved parents Ms. Saida Boureddaya and Mr. Abdelaziz El Alaoui, whom I will never be able thank enough. I thank my sister Ms. Mariam, her husband Mr. Zakaria and lovely daughter Samine, my brother Mr. Youssef and his wife Ms. Sara, and my dear in-laws. I am grateful for my family's love,

pride and support that reach me through the transatlantic telecommunication cables.

Finally, I am most grateful to my dearest, my husband Mr. Baligh Ben Taleb. He has always believed in me and filled me with love, energy and much needed support to overcome the challenges of graduate school and life in general. I am truly blessed to have had him in this adventure, and look forward to many more to come.

I dedicate this work and my successes to family and to the sweetest gift from God, Adam (the little drunken sailor).

Table of Contents

List of Figures **xii**

List of Tables **xiv**

1 Introduction **1**

1.1 Contributions 4

1.2 Organization of the Dissertation 6

2 Heterogeneous Networks Communication Model **8**

2.1 Introduction 8

2.1.1 Fostering Heterogeneity 9

2.1.2 Application Domains 11

2.1.2.1 Small Scale Network Example 12

2.1.2.2 Large Scale Network Example 12

2.2 Heterogeneous Wireless Networks Contact Classification 13

2.2.1 Related Work 14

2.2.2 General Notations 15

2.2.3 Our Proposed Contact Classification 15

2.2.3.1 Definitions 15

2.2.3.2 Discovered contacts (C1) 17

2.2.3.3 Predicted contacts (C2) 19

2.2.3.4 Scheduled contacts (C3) . . . 21
2.2.3.5 Continuous contacts (C4) . . . 23
2.2.4 Heterogeneous Network Generalized Definition . . . 24
2.3 Conclusion . . . 26

3 Single-Attribute and Multi-Attribute Routing and Scheduling for Space Networks 27

3.1 Introduction . . . 27
3.2 Background and Related Work . . . 31
3.2.1 Space Networks . . . 31
3.2.1.1 Routing in Space Network . . . 31
3.2.1.2 Scheduling in Space Networks . . . 33
3.2.2 Multi-Attribute Decision Making (MADM) Optimization . . . 34
3.3 Formal Definition of the Scheduling and Routing Problem in Space Networks . . . 35
3.3.1 General Notation . . . 35
3.3.2 Single-Attribute Scheduling and Routing Model . . . 36
3.3.3 Multi-Attribute Scheduling and Routing Model . . . 41
3.4 N-Look Ahead Routing and Scheduling Algorithm (N-LARS) . . . 42
3.4.1 N-LARS Heuristic . . . 42
3.4.2 Validation and Numerical Results . . . 44
3.4.2.1 Experimental Setup and Network Configuration . . . 45
3.4.2.2 Numerical Results . . . 46
3.5 The Multi-Attribute Routing and Scheduling Algorithm . . . 49
3.5.1 MADM for IPN Routing . . . 49
3.5.1.1 Choice of Attributes and Computation of Their Weights 49

3.5.1.2 PROMETHEE II Applied to Space Networks 50
3.5.2 MARS: the Multi-Attribute Routing and Scheduling Algorithm 52
3.5.2.1 Route-Transmit Algorithm 54
3.5.3 MARS Algorithm Proof and Complexity 54
3.5.3.1 MARS Algorithm Proof 54
3.5.3.2 Complexity Analysis 58
3.6 MARS Experimental Setup and Results 59
3.6.1 CGR Implementation and Parameters 59
3.6.2 Moon-Mars Small Network 61
3.6.2.1 Experimental Setup 61
3.6.2.2 Numerical Results 62
3.6.3 Large-Scale Networks 63
3.6.3.1 ONE Network Setup 63
3.6.3.2 Network Traffic 64
3.6.3.3 Numerical Results and Analysis 65
3.7 Conclusions and Future Work 69

4 STAN+PETRA: A Statistical Analysis Aided Routing Algorithm for QoS in Mission-Driven IoT Networks 73
4.1 Introduction 73
4.1.1 Proposed Approach Overview 75
4.2 Related Work 76
4.3 The Statistical Analysis Framework: STAN 78
4.3.1 The Test Case Network: The oviedo/asturies-er Dataset . . . 78
4.3.2 STAN's Exploratory Analysis 79

4.3.2.1 The Five Number Summary and Box-and-Whisker plots . . . 79
4.3.2.2 Decomposition for Stationarity and Seasonality . . . 80
4.3.2.3 Distribution in the Quantile-Quantile (QQ) and Probability Plots . . . 81
4.3.2.4 Comparative Distribution: The 2-Sample QQ Plot . . . 82
4.3.3 STAN's Model Testing and Selection . . . 82
4.3.3.1 Classical Forecast Methods . . . 83
4.3.3.2 Deep Learning Methods . . . 84
4.3.3.3 Grid Searching Hyperparameters . . . 85
4.3.4 STAN's Forecast Module . . . 86
4.4 The Predictive Routing Algorithm (PETRA) . . . 86
4.4.1 The PETRA Routing Algorithm . . . 87
4.4.2 Route Computation . . . 89
4.4.3 Edge Prediction Integration . . . 91
4.5 STAN+PETRA Experimental Setup and Results . . . 92
4.5.1 Implementation of STAN and PETRA . . . 92
4.5.2 Experimental Setup: oviedo/asturies-er Dataset . . . 93
4.5.3 Numerical results and analysis . . . 94
4.5.3.1 STAN's Forecast Accuracy . . . 94
4.5.3.2 PETRA's Numerical Performance . . . 95
4.6 Conclusions . . . 97

5 A Low-Latency Routing and Edge Computing Framework for Heterogeneous Wireless Mission-Driven IoT 99

5.1 Introduction . . . 99

5.2 Background and Related Work . . . 101
5.2.1 Routing and Computing for IoT Networks . . . 101
5.2.1.1 Routing in Resource Constrained IoT Networks . . . 101
5.2.1.2 Computing in Resource Constrained Networks . . . 101
5.2.2 Age of Information . . . 103
5.3 Network Description . . . 104
5.3.1 Mobile, Delay-Sensitive Sensor-Actuator Network . . . 104
5.3.2 ERGO, an Edge Architecture for Ag-IoT . . . 107
5.4 Network Model Specification . . . 108
5.4.1 Definitions and General Notations . . . 108
5.4.2 Network Model . . . 111
5.4.2.1 Processing Nodes Blocking Probability and Waiting Time . . . 111
5.4.2.2 Age of Information . . . 114
5.4.2.3 Node Energy Availability . . . 115
5.4.2.4 Link Blocking Probability . . . 116
5.4.3 Routing dependent variables . . . 118
5.5 Model solving . . . 118
5.5.1 Arrival Rate $\psi_s(X_{t_i}, P_V, u)$ . . . 120
5.5.2 Arrival Rate $\psi_n(X_{t_i}, P_N, n)$ . . . 121
5.5.3 Energy Consumption . . . 122
5.5.4 The Simplified Model . . . 123
5.6 Experimental Setup and Results . . . 124
5.6.1 ERGO Performance Measurement . . . 124
5.6.2 The Low-Latency ILP Routing Evaluation . . . 125
5.7 Conclusions and Future Work . . . 127

6 Conclusions and Future Directions **129**

6.1 Conclusions 129

6.2 Future Directions 131

Bibliography **133**

List of Figures

1.1 Dissertation Organization. 7

2.1 Heterogeneous Wireless Network. 9
2.2 Network contact types. 14
2.3 Temporal Graph and model notation. 15
2.4 Illustration of a network composed of discovered contacts. 17
2.5 Illustration of a discovered network. 18
2.6 Illustration of a network composed of predicted contacts. 20
2.7 Illustration of a network composed of scheduled contacts. 21
2.8 Typical satellite's trajectory to Mars shown in green (Source: [16]). . . . 21
2.9 Illustration of a network composed of all four types contacts. 24

3.1 An illustration of the Space Network (source: [1] - Modified) 28
3.2 Illustration of a Mars-Earth small network used in the experiment. . . . 45
3.3 N-LARS heuristic vs. CGR experimental results. 47
3.4 Preference value intervals. 56
3.5 Experimental network topologies. 60
3.6 Earth-Moon-Mars network performance. 62
3.7 Performance analysis and comparison in different network sizes with $\lambda = 15$. 66
3.8 All network sizes comparison using config2 and λ=15. 67

3.9 Comparison between 255 different MADM weight matrices. 70

4.1 Fire department disaster response network. 74
4.2 System architecture: interaction between STAN and PETRA. 74
4.3 Temporal portioning of oviedo/asturies-er traces. 78
4.5 Additive decomposition of oviedo/asturies-er traces and samples. . . . 80
4.4 The box-and-whisker plot comparing three months. 81
4.6 QQ-plot and the probability plot of one week of traces. 82
4.7 2-Sample QQ plot of one month and one week of traces. 83
4.8 Autocorrelation of a day of data. 83
4.9 RMSE of the four models. 93
4.10 CNN and GBC-adjusted one hour forecast for one pair of nodes. 94
4.11 Daily Throughput. 95
4.12 End to end delay distribution. 96
4.13 Performance metrics summary. 97

5.1 Illustration of a mega-farm monitoring Ag-IoT network. 99
5.2 Network Topology. 100
5.3 Scenario: Tractor with fertilizer sprayer. 104
5.4 ERGO architecture. 106
5.5 Performance evaluation of the Ag-IoT Application APIs. 124
5.6 Example experimental network. Green circle indicates ERGO clusters. . 125
5.7 Performance evaluation of $COTA^u$. 126
5.8 Traffic delivery and processing. 126

List of Tables

3.1 Formulation parameters 36
3.2 Formulation variables 37
3.3 Extra Formulation parameters 41
3.4 Uplink and downlink data rates in Kbps 62
3.5 Examples of network scenarios from real missions (The number of objects are not scaled because the missions are still limited in the deep space). 63
3.6 Summary of window size for maximal bundle delivery in each network and traffic load. 65
3.7 Weight matrix configurations for the higher delivery ratio. 69

4.1 Variables used in Algorithm 5. 90
4.2 Methods used in Algorithm 6. 91

5.1 Low Latency Routing Model parameters 111

Chapter 1

Introduction

As the number of connected devices is increasing, the network architecture is constantly shifting towards more heterogeneity and flexibility in communication media and standards to incorporate the diversity in devices and technologies they use to operate and communicate. Different network applications have different requirements with regard to communications, quality requirements, and devices and technologies compatibility. For example, consider a smart home; there are devices from different vendors, with different enabling technologies, various requirements, and competing needs for resources. This example and many others fall under the umbrella of heterogeneous networks because the term heterogeneous can take many definitions as detailed in the technical report on Future Heterogeneous Networks [2]. At a time where smart communication technologies are set to solve many of our problems and make our lives easier and better, it is important to design and implement network architectures that are robust to heterogeneity.

A network is heterogeneous if it has diversity in one or more of the following aspects:

- **Technologies:** The use of multiple communication technologies, such as

WiFi, 4G, 5G, Bluetooth, etc., impacts compatibility and hinder communications between network entities using different technologies.

- **Devices:** Heterogeneous devices that are manufactured by different vendors and run various operating systems, which results in a diversity of hardware and software compatibility limitations in terms of protocols, ease of integration of new frameworks, upgrade and adaptability to new technologies and so on.

- **Transmission media:** Heterogeneous transmission media are manifested in the use of different technologies such as satellite communication, free space optical links, terrestrial wireless links, underwater wireless links, Visible Light Communication (VLC) links and so on [2].

- **Mobility:** Heterogeneous mobility models are present when the network is composed of devices that are fixed and others that are mobile and that could follow different mobility models.

- **Communication protocols:** Heterogeneous communication protocols can be especially challenging and require the use of gateways or frameworks capable of seamlessly understanding and processing data streams from different protocols used by similar or varying communication modalities.

- **Channel types:** Heterogeneous channel types are defined in [2] as "the rate of change and predictability of change of channel characteristics," which can be measured by the channel's bandwidth, data rate, bit-error rate (BER), utilization, signal-to-noise ratio (SNR) and latency among others. These characteristics are defined to a large extent by the communication protocol, transmission media and mobility.

- **Traffic:** Heterogeneous traffic types which are usually described using the four V's: Velocity, Veracity, Volume and Variety (Value is sometimes included as the fifth V).

- **Quality requirements:** Heterogeneous applications that use the same network architecture require the support for various quality requirements and dimensions, such as Quality of Service (QoS), Quality of Experience (QoE), Quality of Data (QoD) and Quality of Information (QoI). Quality dimensions are delineated by resilience and robustness of the network, which are in turn bound to constraints including, and not limited to, transmission media, communication protocols and mobility models.

- **Management and control:** Heterogeneous "network management and control systems and administrative domains" [2] are manifested, for example, in the choice of distributed or centralized network control. Inter-domain and intra-domain management systems have been widely studied, but that constitute a different challenge when the network has other heterogeneity aspects.

Current research efforts attempt to solve the bigger problem of heterogeneity by making assumptions about the network that make the problem less complicated; that is, only selected aspects of heterogeneity are included. Another approach is generalizing the problem to include most of the heterogeneity aspects while compromising the solution's applicability to any specific application. For example, a solution that is designed when considering constraints of a disaster recovery network might not perform as well when implemented in an agricultural IoT network. While overlooking the importance of application specific requirements, the proposed solutions are not immediately applicable to a given prob-

lem/situations. Hence minor, and sometimes significant, changes would have to be made in order to apply these solutions to any specific application. In order to overcome these limitations, these networks should be properly modeled to a level of detail that reflects each aspect of heterogeneity. Such elaborate network models would facilitate the creation of solutions that fulfill the requirements of applications without overlooking the effects of heterogeneity on the overall performance of the network architecture.

1.1 Contributions

The research presented in this dissertation is on the overarching topic of routing and optimization in heterogeneous networks that are deployed in challenged environments, such as Interplanetary networks and disaster recovery networks. We study these networks based on their contact types, which allow the extraction and modeling of the main heterogeneity features of these networks. Hence, in **Chapter 2**, we developed a novel and comprehensive mathematical model for heterogeneous networks' contact classification composed of four main categories, namely continuous, scheduled, discovered and probabilistic. We hence investigate questions pertaining to routing and optimization in networks of each particular category of contacts and propose a generalized framework for routing in heterogeneous networks composed of multiple or all of those contact categories.

We start by studying scheduled contacts in Interplanetary Networks (IPN). IPN are on the spectrum of challenged networks with high communication delays; therefore, there are designated as Delay Tolerant Networks (DTN). For this work, we propose a novel Modified Temporal Graph (MTG) model to represent Delay Tolerant Interplanetary Networks in a near-real-time deterministic dynamic repre-

sentation. This allowed us to design and implement a novel Earliest Arrival Optimal Delivery Ratio (EAODR) routing algorithm. The use of the MTG model along with EAODR helped reduce the end-to-end communication delay. To further increase the throughput of the network at a reduced end-to-end communication delay, we propose the first Mixed Integer Linear Programming (MILP) model for message routing and scheduling in the IPN using Multi-Attribute Decision Making (MADM) principles. We then design and implement a novel MADM-based algorithm called Multi-Attribute Routing and Scheduling (MARS) algorithm that uses a sliding window to schedule messages and EAODR for route computation. We run experiments on a real-world network featuring Mars lander and orbiters, and we further evaluated our work on large networks that are composed of real space objects from AGI's Space Tool Kit (STK). Results from this work have been published in the form of both conference papers [3–5] and journal articles [6, 7], and are detailed in **Chapter 3**.

In studying heterogeneous networks that have all types of contact, we focus on heterogeneous wireless Mission Driven IoT (MD-IoT) and construct a comprehensive mathematical model, based on which we designed and implemented a Statistical Analysis (STAN) framework to statistically analyze the MD-IoT network traces and predict future traces that can be used for route computation. The STAN framework allows nodes in the network to choose an appropriate forecast model for future contact prediction. The framework incorporates a variety of deep learning or traditional forecast models enhancing the flexibility of the framework in adapting to network application and communication traces. We designed and implemented the predictive routing algorithm (PETRA) that uses STAN's predictions in route computation and run network simulations using a dataset composed of real traces collected from a Fire Department. Preliminary

results from the effort we presented at the IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS) [8] and are presented in **Chapter 4**.

The last part of this research is focused on delay-sensitive agricultural IoT networks that require near real-time data processing, communication and interventions. This work focuses on a network composed mainly of predicted and discovered contacts, along with multiple aspects of heterogeneity in data types, equipment, communication media and so on. In this work, we investigate techniques that expedite the transfer of data to processing nodes in the network along with optimization in processing nodes positioning and computation. We are working on the design of network models for agricultural applications, which will allow us to implement efficient routing algorithms integrated with edge computing solutions for near-real time data processing. The work will then be evaluated using data collected by collaborators at UNL's College of Agricultural Sciences and Natural Resources. Details of this ongoing work is presented in **Chapter 5**.

1.2 Organization of the Dissertation

This dissertation presents our four main contributions. We start by creating and formalizing the classification model for heterogeneous wireless networks based on the four types of contacts that identify as: scheduled, predicted, discovered and continuous. As depicted in Fig. 1.1, The model is introduced in **Chapter 2**, which is integral to the other three contributions. **Chapter 3** start by presenting out work on single-attribute and multi-attribute message scheduling and routing for scheduled contacts using space networks as an example. We then consider networks composed of all four types of contacts in MD-IoT environments and

propose s statistical analysis-backed predictive routing algorithm in **Chapter 4**. Last but not least, **Chapter 5** shown in Fig. 1.1, studies MD-IoT networks that are composed of predicted and discovered contacts, for which we propose a low-latency routing algorithm along with edge computing capabilities.

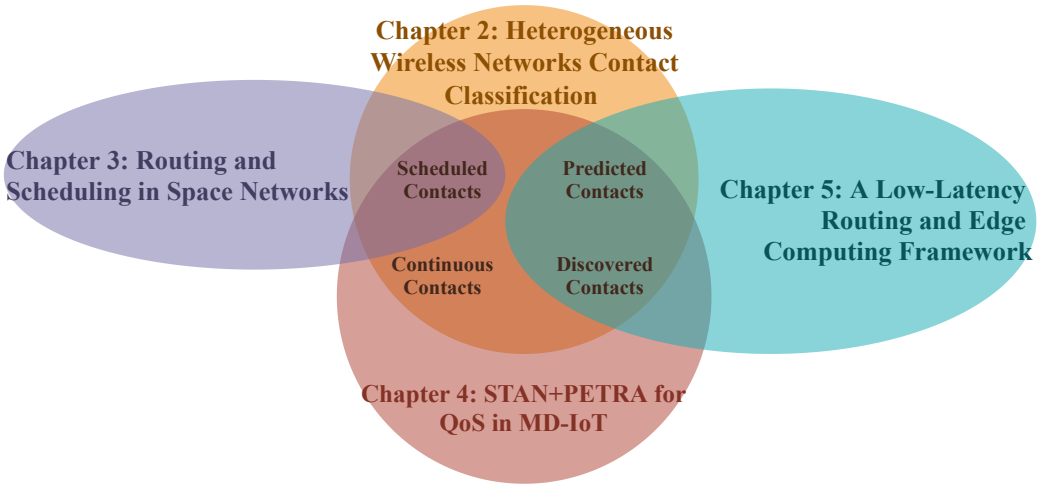


Figure 1.1: Dissertation Organization.

Chapter 2

Heterogeneous Networks Communication Model

2.1 Introduction

Heterogeneous networks are networks that are composed of devices that either have different capabilities or operate on different levels of requirements and protocols. The majority of devices around us are capable of sensing data from the environment and sharing it with other devices. In order for this communication to take place, the network should be designed to seamlessly support a variety of devices, protocols and applications. And this becomes possible by studying heterogeneous networks and designing architectures that are robust to heterogeneity. One manifestation of heterogeneity of networks is the diversity of contacts established for communicating links at different points in time and space, where a contact is a communication window between two network entities. In this chapter, we reiterate the importance of heterogeneous networks, providing examples of application domains. Then, propose a novel classification of contacts in these networks, which constitute the basis for a network architecture.

2.1.1 Fostering Heterogeneity

In attempting to reduce heterogeneity and ensure compatibility between emerging technologies, agencies have led efforts to create standards for compliance. Past and ongoing standardization processes try to discern different technologies from each other, and provide definitions and compliance rules; however, the pace at which new technologies are released to the end user is much higher than what it takes to come up with the standard that fits all varieties of the same invention. That is also because many innovations are created by breaking from the initial standards. Different devices also generate various types of traffic; a heterogeneity aspect that cannot easily be limited by standards but that is highly affected by the latter. That is, standards used for other aspects such as communication protocols can greatly shape the network traffic flows. Regardless of all these differences and implementation variations, the vast majority of the applications running over the network are constantly evolving and collecting data from the end users (images, videos and so on) that are to be shared or processed in the Cloud or other (local or remote) facilities.

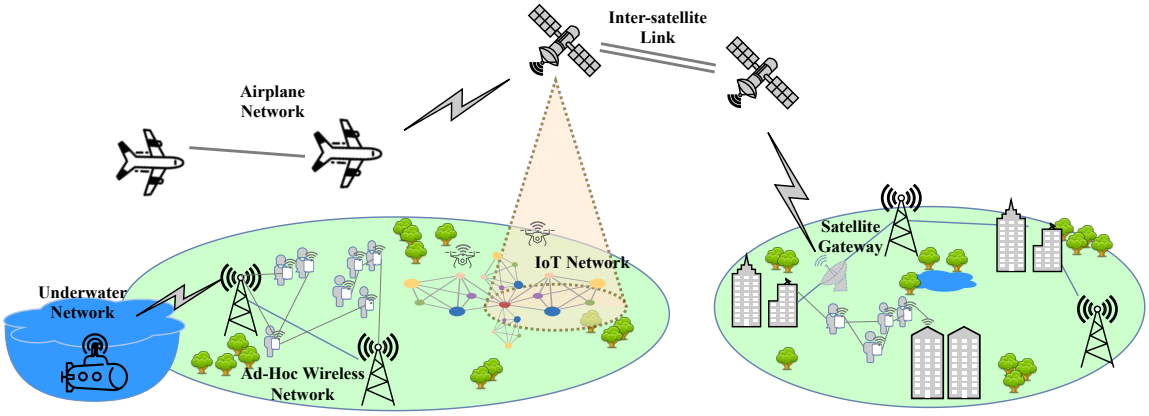


Figure 2.1: Heterogeneous Wireless Network.

Figure 2.1 shows a representation of heterogeneity in our everyday networks; it depicts the aspects discussed earlier as they are related to technologies used in

real networks. As depicted, a network can be composed of satellite links through gateways with higher delays; airplane networks that connect through satellite links and rely on inter-satellite links as well; underwater networks that report data back to the surface; and other ad-hoc wireless and IoT networks. All these networks are interconnected in various aspects and introduce heterogeneity at various levels listed in Chapter 1. While heterogeneity presents several implementation challenges, there are many advantages to designing networks with an inherent support for heterogeneity. We elaborate on some of them in the following.

- **Scalability**: Networks that are designed with homogeneous nodes in all aspects mentioned in Section 1 can be hard to expand. Any new device that is connected to this network would have to fit a large set of requirements, limiting the options and possibly even increasing the cost of scaling up. In a homogeneous wireless sensor network, for instance, any new sensors and gateway nodes would have to have the exact same specifications as the ones already deployed in the network, which removes the flexibility of integrating new devices from various vendors. This could make the price higher, both in terms of money and operations.

- **Innovation in technology**: The inherent support for heterogeneity alleviates the burden of compatibility and hence permits more freedom to invent and innovate. When the network architecture is built on heterogeneity, innovative research is not bound by the software, hardware and communications media compatibility requirements.

- **Novel applications**: The efficient integration of novel applications is facilitated by the use of heterogeneous network architectures and the fact that

they benefit from scalability and ease of innovation. As mentioned earlier, for various applications to appropriately function on the network, the latter has to support various quality requirements and domains and diverse performance levels, and it should simplify the integration of new equipment, protocols and communication media. Therefore, the integration of novel applications is greatly incumbent upon mechanisms that account for heterogeneity in the network.

- **Resilience and robustness through diversity**: For all the aspects of heterogeneity mentioned in Chapter 1, its various ways of implementation have distinct set of limitations and advantages. And one option's limitations can be addressed by another option's advantages. For example, while fixed sensors can only operate in a specific limited area with strict communication constraints, mobile sensors (e.g. UAVs) can compensate for the spatial limitations of their fixed counterparts. Likewise, using a diversity of communication media incorporates network resilience and robustness. For instance, when a satellite communication if compromised by weather conditions, other communication media can be used to build a backup communication path until the satellite link is restored.

2.1.2 Application Domains

Applications that seemed hard to achieve a decade or two ago are becoming feasible thanks to the development of technology and the great progress that innovations in satellites, sensors and Unmanned Aerial Vehicles (UAV) have shown. Advances in ground terminals and antennas, reusable and sustainable space technology, payload augmentation, autonomous UAVs, and especially the ability to

integrate all this diversity in heterogeneous network architecture, have opened the door for new network applications and scenarios that were beyond reach. Some of these applications are introduced below.

2.1.2.1 Small Scale Network Example

Disaster management and recovery is a powerful example of heterogeneous networks. Aeronautical communication technologies are of great importance in operating a network whose infrastructure is fully or partially damaged. Satellite, UAVs and other mobile objects can help during the emergency relief process, making it smoother, more efficient and safer for the rescue teams. As such, these networks involve different types of communicating nodes with various capabilities and constraints, and on top of that they are usually deployed in an ad-hoc manner. Establishing such a network architecture should be optimized by seamlessly integrating heterogeneity to build a robust and reliable network.

2.1.2.2 Large Scale Network Example

On a different scale and in preparation for deep space missions, space research centers are working on deploying networks of robots on the surface of the moon (i.e. cis-lunar space) in order to assess and verify the technologies and equipment before deploying them on farther planets building Interplanetary Networks (IPNs) [9, 10]. Both missions anticipate deploying habitats for humans in addition to rovers and robots. The mission crew will then be able to remotely operate the rovers and robots either from the deployed habitats or from the surface of Earth [11]. These network setups, cis-lunar networks and IPNs, are unique environments where there are objects that follow well-defined schedules of communication relative to their line-of-sight windows (e.g. satellites), and objects that can

be reached in an ad-hoc manner, unless they follow predetermined trajectories (e.g. robots and rovers) in addition to objects whose position can be predicted following their trajectory. Besides their varying mobility models, these objects have various quality requirement, communication capabilities, data flow types and so on.

As these examples of applications reiterate the relevance of heterogeneity in our daily life, they stress the importance of fully understanding the dynamics of such networks in order to better design and implement architectures capable of making use of heterogeneity to its full potential. With that said, we study scheduled contacts using the Interplanetary Network as a use-case. Networks composed of all types of contacts are then investigated with a focus on the Mission-Driven IoT networks. Finally, discovered and predicted contacts are then studied in the Agricultural IoT networks. In the following section, we provide a novel general model that encapsulates different aspects of heterogeneity listed in Chapter 1. The model is based on the temporal graph model that supports spatio-temporal network configurations [6].

2.2 Heterogeneous Wireless Networks Contact Classification

An important step in solving heterogeneous network design and research questions is to fully understand their dynamics. In the remainder of this chapter, we present a heterogeneous network model that allows to capture and understand various aspects of heterogeneity. This model provides the basis to which details can be added for specific research questions. The model is based on defining four types of contacts (i.e. time slots during which two nodes in the network can exchange data) and a temporal graph. Together, they encompass the heterogeneity

of a network.

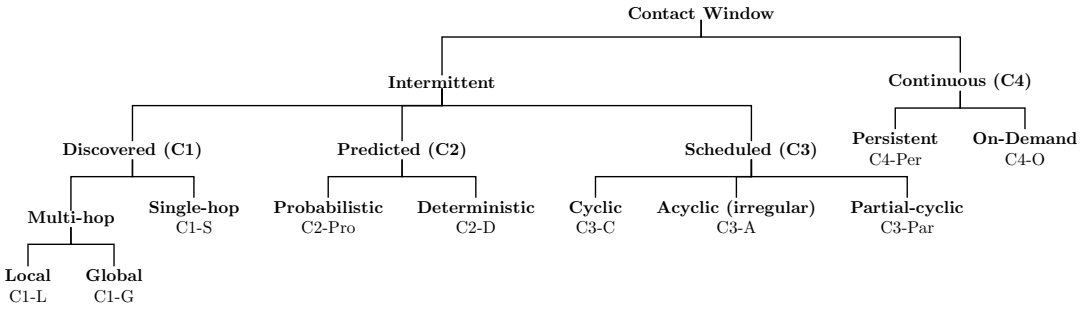


Figure 2.2: Network contact types.

2.2.1 Related Work

RFC4838 [12] was one of the first documents to provide a definition of types of contacts in the context of the Delay Tolerant Networking architecture. It defined five types of contacts: Persistent, On-demand, Scheduled, Opportunistic, and Predicted; all defined in high level definitions. Morgenroth et al. [13] have reduced the number of categories of contacts to four by removing On-demand contacts from their categorization. They have also defined two types of scheduled networks. They introduced Hard-scheduled and Soft-scheduled, which refer to contacts with well known times and those that are known but with uncertainty, respectively. They classify these categories in a scale ranging from high confidence to high uncertainty. Similar definitions have been provided in [14], providing high level terms and focusing on examples to better illustrate the differences. Several other definitions were proposed based on these baselines; however, they all have overlaps between different categories creating confusion and ambiguity in the understanding of the network dynamics. Furthermore, these definitions do not provide enough details on each category that facilitate building elaborate and detailed network models.

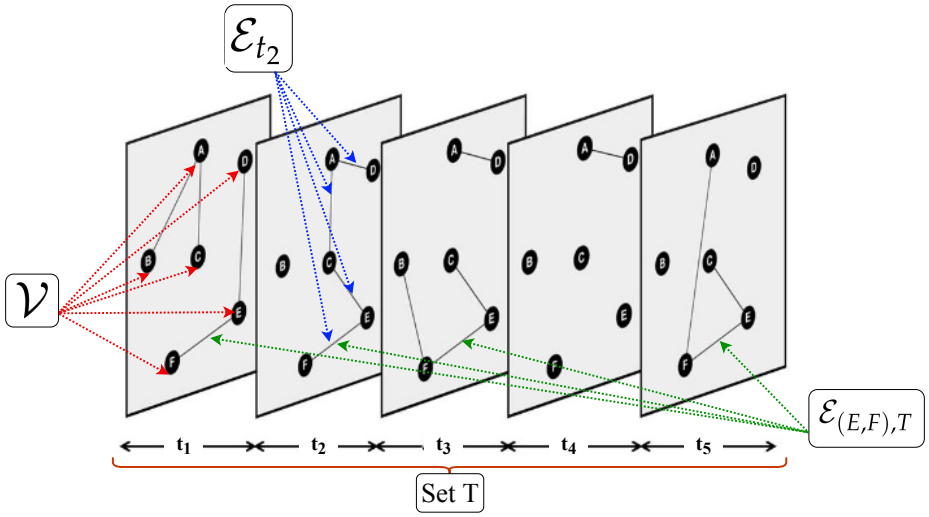


Figure 2.3: Temporal Graph and model notation.

2.2.2 General Notations

The network will be represented as a temporal graph that we denote by G_T, as depicted in Fig. 2.3. The temporal graph is characterized by a set of temporal edges, $\mathcal{E}_T$, where an edge with nodes (u, v) has multiple occurrences with different start times and durations. Hence we write $G_T = (\mathcal{V},\ \mathcal{E}_T)$.

2.2.3 Our Proposed Contact Classification

2.2.3.1 Definitions

The notation that will be used in the formal definitions is introduced bellow and depicted in Fig. 2.3.

- Vertex/node: A vertex (a.k.a. node) is defined as a data structure that contains information such as storage, processing capabilities and so on. It is generally denoted by u, v, a, or b with subscripts when needed.

- $e_{(u,v),t}$: The temporal edge between nodes u and v during time slot t.

- Temporal edges: Temporal edges e_i has to store at least four pieces of information as $e_i = (t_s, \Delta t, r_{up}, r_{down})$: 1) its start time t_s, 2) its duration Δt, and 3) its uplink data rate and 4) downlink data rate, r_{up} and r_{down} respectively. Additional information can be added to the data structure as needed.

- $\mathcal{E}_{(u,v),T}$ (also $\mathcal{E}_{(u,v),t}$ for simplicity) is the set of temporal edges between nodes u and v and is defined as: $\mathcal{E}_{(u,v),T} = \{e_{(u,v),t} | \forall t \in T\}$.

- $\mathcal{E}_t$ is the set of all the temporal edges between all pairs of nodes that are present in the network in the time slot t.

- T is the set of time slots.

- $\mathcal{E}_T$ is the set of all the temporal edges between all pairs of nodes that are present in the network throughout all the network's lifetime.

- $\mathcal{E}_{u,t}$ is the set of all temporal edges known to node u. These could include contacts between u and other nodes, as well as contacts between different nodes in the networks. We may also use a superscript to indicate the category of contacts being defined.

As shown in Fig. 2.2, we define two major types of contacts: *Intermittent* are discontinuous and occur at different times with, usually, a different duration each time. Therefore, for a certain time duration, there are multiple temporal edges between every pair of nodes, and *Continuous* where the pair of node is constantly connected; therefore, there is only one continuous edge between then. All other categories are detailed next.

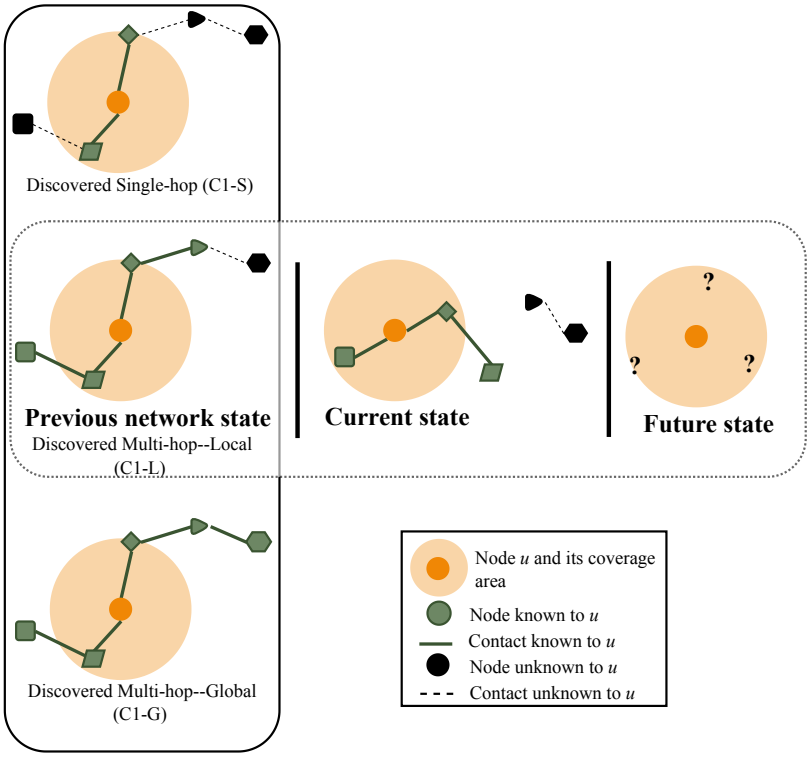


Figure 2.4: Illustration of a network composed of discovered contacts.

2.2.3.2 Discovered contacts (C1)

Discovered contacts (C1), illustrated in Fig. 2.4, are ones where the motion of the nodes is random, and hence, no prior knowledge of contact windows is available. We define the mathematical model for this category of contacts as follows.

Given a node $u \in \mathcal{V}$ at time t, the neighborhood of u is the set of discovered nodes:

$$\mathcal{N}_{u,t} = \{v_1, v_2, ..., v_{m_t} \,|\, \forall i \in (1, m_t), v_i \in \mathcal{V}\}$$

Each of the nodes $v_i \in \mathcal{N}_{u,t}$ are defined as the triplet $v_{k,(i,t)} = (d_k, v_k, \varphi_k)$ where d_k is the initial distance between v_i and $v_{k,(i,t)}$ at time t. v_k is the velocity of $v_{k,(i,t)}$, and φ_k is its angle. Nodes can use different equation, such as the one in [15]and as depicted in Fig. 2.5, to compute the contact duration.

We then define the set of edges for a node $u \in \mathcal{V}$ at time slot $t \in T$ as follows:

$$\mathcal{E}_{u,t}^{\mathcal{C}1} = \alpha \left(\bigcup_{v \in \mathcal{N}_{u,t}} \mathcal{E}_{v,t} \setminus \mathcal{E}_{(u,v),t} \right) \bigcup \left(\bigcup_{v \in \mathcal{N}_{u,t}} \mathcal{E}_{(u,v),t} \right) \tag{2.1}$$

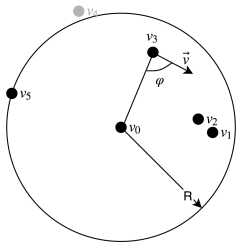


Figure 2.5: Illustration of a discovered network.

The first part of the equation allows node u to include to its contact list all contacts that have been discovered by all its neighbors in the set $\mathcal{N}_{u,t}$, subject to the function α, which we define as part of each subcategory of the Discovered Contacts (C1). The second part of set definition defined the set of all the temporal edge between the node u and all its neighbors within the time slot t. Equation (2.1) defines the Discovered Contacts that are further broken into two main subcategories as follows.

- **Multi-hop**: which means the node can discover contacts in the network beyond its immediate neighbors, and it is further categorized into two types.
  - Local (C1-L): In situations where the node has constraints of any types such as energy or storage, it is not practical for node u to save all shared contacts by all its neighbors in the set $\mathcal{N}_{u,t}$; therefore, only local knowledge of the network is required. In such case, we use the constraint function α to limit the contact sharing with neighbors. The constraint could be a limit on the distance between node u and any other node a ($dist(u,a)$), a threshold on the data rate of the link and so on. For example, $\alpha(X) = \{e_{(u,v)} | \forall u, v \in X \; s.t. \; dist(u,v) < d\}$, set a constraint on the minimum distance between two nodes to be included in the set X.
  - Global (C1-G) where nodes use a mechanism, to share their whole list

of discovered contacts without any constraints. Therefore, we define α as $\alpha(X) = X \cup \emptyset$. We hence write:

$$\mathcal{E}_{u,t} = \bigcup_{v \in \mathcal{N}_{u,t}} \left(\mathcal{E}_{v,t} \setminus \mathcal{E}_{(u,v),t} \right) \bigcup \left(\bigcup_{v \in \mathcal{N}_{u,t}} \mathcal{E}_{(u,v),t} \right) \tag{2.2}$$

- **Single-hop** (C1-S) contacts occur in a network where the node is only able to discover its immediate neighbors. Therefore, it is a special case of the multi-hop network where α is defined as $\alpha(X) = X \cap \emptyset$. $\mathcal{E}_{u,t}$ is defined as follows:

$$\mathcal{E}_{u,t} = \bigcup_{v \in \mathcal{N}_{u,t}} \mathcal{E}_{(u,v),t} \tag{2.3}$$

2.2.3.3 Predicted contacts (C2)

Predicted contacts (C2) are contacts between nodes that follow a specific motion pattern or have information of each others' mobility variables such as trajectory, speed and acceleration. Examples of these networks are satellite networks and vehicular networks respectively. There are also predicted contacts that are derived from prior knowledge of the network. Therefore, using mathematical equations or knowledge of the network, a node u can generate the set of temporal edges $\mathcal{E}_{u,t}$. As shown in Fig. 2.6, we define two types of predicted networks.

- **Probabilistic** (C2-Pro): Probabilistic contacts are contacts that are predicted but have a probability $\beta < 1$ of happening. These contacts are prevailing in bus (vehicular) networks for example, where we know with a certain probability that the bus will arrive within a time interval.

- **Deterministic** (C2-D): The deterministic contacts on the other hand are ones where the predicted contact is guaranteed to happen at that specific time

with a probability $\beta = 1$.

Every node u contains a set of encountered nodes, $\mathcal{O}_{u,t}$, which includes nodes that have been temporal neighbors of u and others that it has discovered through the Multi-hop discovered contact categories $C1 - L$ and $C1 - G$. The encounter set is defined in Eq. (2.4).

$$\mathcal{O}_{u,t} = \{v_1, v_2, ..., v_{c_t} \,|\, \forall i \in (1, c_t), v_i \in \mathcal{V}\} \tag{2.4}$$

We also define the set $\mathcal{P}_v = \{p_1, p_2, ..., p_l\}$ representing all the properties of node v, and which can include node u's prior knowledge of v mobility; the latter's velocity, angle, and initial position; or any other information used for the prediction. We also define Eq the set of equations used by a function f that takes as input source node v_s, destination node v_d, its set of properties $\mathcal{P}_v$, and returns its predicted temporal contacts, the edge $e_{u,v}$ with a probability β. Hence the set $\mathcal{E}_{u,t}$ for predicted contacts is defined as:

$$\begin{aligned} \mathcal{E}^{C2}_{u,t} = \{f : v_s,\, v_d,\, \mathcal{P}_{dst},\, Eq \rightarrow\, & (e_{(v_s,v_d),t},\, \beta) \\ & |\forall v_s, v_d \in \mathcal{O}_{u,t} \cup \{u\}, v_s \neq v_d\} \end{aligned} \tag{2.5}$$

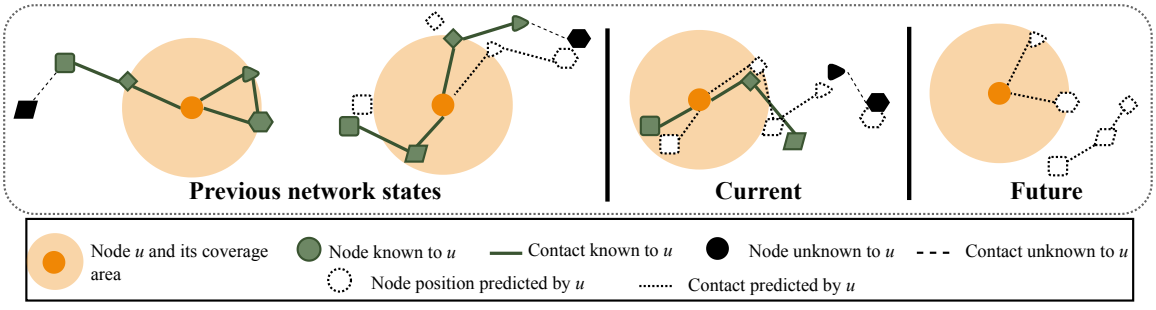


Figure 2.6: Illustration of a network composed of predicted contacts.

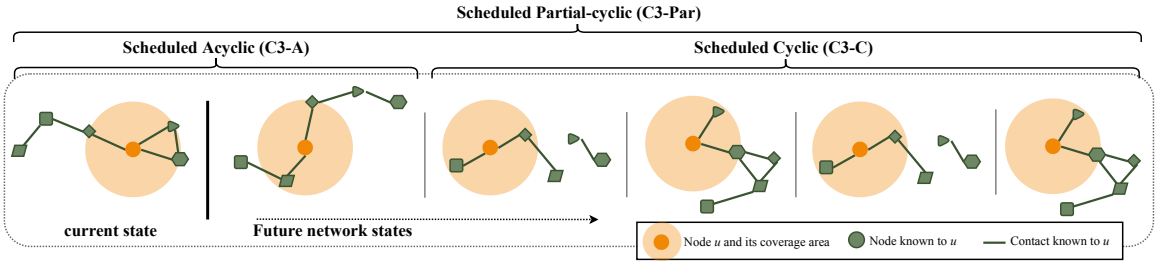


Figure 2.7: Illustration of a network composed of scheduled contacts.

2.2.3.4 Scheduled contacts (C3)

Scheduled contacts (C3), depicted in Fig. 2.7, are at the other extreme where the schedule is pre-loaded in each node as part of the mission planning and operation process. There are three sub-types of contacts.: *cyclic*, *semi-cyclic* and *acyclic*. The set $\mathcal{S}$ represents the set of all the nodes for which the schedule of contacts is provided, and we call them scheduled nodes for convenience. The edge set $\mathcal{E}_{u,t}$ would be the same for all time slots $t \in T$ since the nodes has full knowledge of the network at all times. We hence write it as:

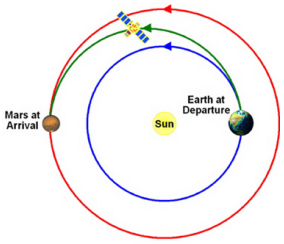


Figure 2.8: Typical satellite's trajectory to Mars shown in green (Source: [16]).

$$\mathcal{E}^{C3}_{u,t} = \bigcup_{\substack{v_i, v_j \in \mathcal{S} \\ v_i \neq v_j}} \left(\bigcup_{\tau \in T} \left(\bigcup_{t \in \tau} e_{(v_i, v_j), t} \right) \right) \tag{2.6}$$

The set $\mathcal{S}$ would include node u is the latter is a scheduled node. The time window τ will be defined in the following and T is, as defined earlier, the set of all time slots/windows constituting the whole time span of the network. The three sub-categories of scheduled contacts are defined next.

- **Cyclic** (C3-C): In a constellation of Medium Earth Orbiting (MEO) satellites,

every node (satellite) follows an orbit that does not change in cycle that repeats itself every τ hours for example. In this setup, every pair of scheduled nodes, v_i and v_j, that come in line of sight will establish a contact e with a duration Δt that will be repeated depending on their respective orbital information. Therefore, T is divided into equal time windows equal to the cycle of duration τ. Edge within the cycle τ will occur at times slots t and will be repeated in the next cycle. Consequently, the set of all edges is composed of edges $e_{(v_i,v_j),t}$ for all time slots t in τ, and all these edges repeated in all cycles τ in the network time span T as shown in Equation (2.6).

- **Acyclic** (C3-A): An example of this is a mission where two nodes can only communicate with each other in pre-scheduled allotted time slots such as a satellite communicating with a base station on the surface of Earth but that is only being used in allotted time slots. This is defined as special case where T is composed of one cycle that is equal to the whole time span. Using Equation (2.6), we set $T = \tau$ and use the same formal definition. Equation (2.6) is modified as follows:

$$\mathcal{E}_{u,t}^{\mathcal{C}3} = \bigcup_{\substack{v_i,v_j \in \mathcal{S} \\ v_i \neq v_j}} \left(\bigcup_{t \in T} e_{(v_i,v_j),t} \right) \tag{2.7}$$

- **Partial-Cyclic** (C3-Par): The partial cyclic contacts are contacts that are cyclic in portions of the lifetime of the network and are acyclic in others. For example, while a satellite is traveling to Mars (as shown in Fig. 2.8, the Satellite-Earth contact is acyclic; however, once the satellite is in orbit, the contact becomes cyclic. The model used for these contacts is based on Equation (2.6) as well. The difference is that T is now written as $T = T_a \cup T_c$. T_a is

the set of acyclic time slots, and T_c is the set of cyclic time slots. Hence, its model is a combination of the models for categories C3-C and C3-A:

$$\mathcal{E}_{u,t} = \begin{cases} \bigcup_{\substack{v_i,v_j \in \mathcal{S} \\ v_i \neq v_j}} \left(\bigcup_{\tau \in T_c} \left(\bigcup_{t \in \tau} e_{(v_i,v_j),t} \right) \right) \\ \bigcup_{\substack{v_i,v_j \in \mathcal{S} \\ v_i \neq v_j}} \left(\bigcup_{t \in T_a} e_{(v_i,v_j),t} \right) \end{cases}$$

Further details about scheduled networks modeling and routing are studied in [6].

2.2.3.5 Continuous contacts (C4)

There are two main types of continuous contacts as defined in [12].

- **Persistent** (C4-Per): These are contacts that are always up and available. That is, "no connection-initiation action is required to instantiate a persistent contact" [12]. Contacts with servers on the Internet are an example, such that the server is always up and waiting for a request.

- **On-Demand** (C4-O): This is a special case of the persistent contacts where the contact is established when it is requested, then it behaves as a persistent contact until it is terminated by one of the parties [12].

Both categories can be modeled using Equation (2.6) similar to the Scheduled Acycli contact (C3-A). We defined the the set $\mathcal{R}$ as the set of all the persistent nodes (equivalent to the set $\mathcal{S}$), and we write:

$$\mathcal{E}^{\mathcal{C}4}_{u,t} = \bigcup_{\substack{v_i,v_j \in \mathcal{R} \\ v_i \neq v_j}} \left(\bigcup_{t \in T} e_{(v_i,v_j),t} \right) \tag{2.8}$$

In order to model networks that are composed of more than one of the four contact categories, discovered, predicted, scheduled and continuous, it is important to define a network model that incorporates all four contact categories. This is defined in the next section.

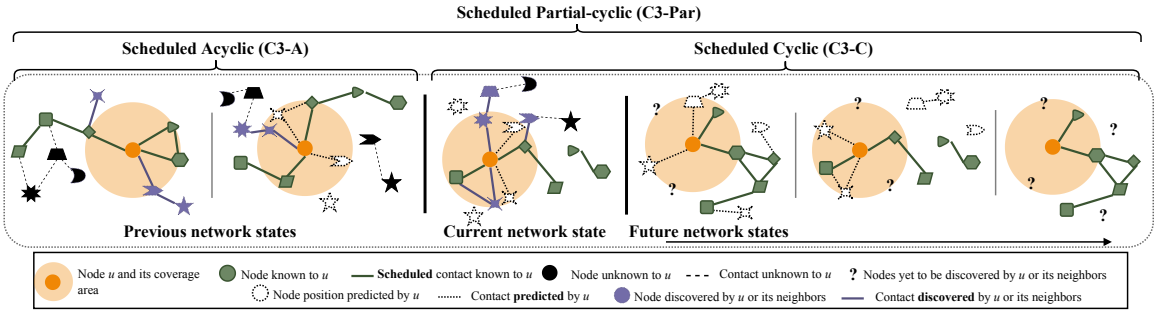


Figure 2.9: Illustration of a network composed of all four types contacts.

2.2.4 Heterogeneous Network Generalized Definition

By definition, a heterogeneous network will be composed of different types of contacts; thus, in order to model these networks, the model should be derived from the equations presented in Section 2.2.3 using all four types of contacts as shown in Fig. 2.9. In the details in the latter, we group the contacts into two groups of models: 1) Discovered (C1) and Predicted (C2) and 2) Scheduled (C3) and continuous (C4). Prior to discovering the C1 contacts, they are added to the contact list as Predicted contacts (C2). That is, as the node u predicts a contact e_i with another node v, the contact is only use if it actually occurs, which then becomes a C1 contact with the same properties (depending on the accuracy of prediction). The second group is complementary and can be define using the dame mathematical equation by varying the temporal aspect of it. In other words, the difference is in the definition of T, τ and t. Accordingly, the general notation

for heterogeneous networks definitions is as follows.

$$\mathcal{E}_T = \lambda\left(\mathcal{E}_t^{\mathcal{C}1},\ \mathcal{E}_t^{\mathcal{C}2}\right) \bigcup \mathcal{E}_t^{\mathcal{C}3} \bigcup \mathcal{E}_t^{\mathcal{C}4} \tag{2.9}$$

In Equation (2.9), we use $\mathcal{E}_t^{\mathcal{C}i} = \cup_{v \in \mathcal{V}} \mathcal{E}_{v,t}^{\mathcal{C}i}$ with its definition in Sec. 2.2.3.1. The function λ is defined to allow us to replace the predicted contacts by their discovered counterparts and is defined as:

$$\lambda\left(\mathcal{E}_t^{\mathcal{C}1},\ \mathcal{E}_t^{\mathcal{C}2}\right) = \begin{cases} e^{\mathcal{C}1} & , e_i^{\mathcal{C}1} \approx e^{\mathcal{C}2} \\ e^{\mathcal{C}1}\ \&\ e^{\mathcal{C}2}, & \textit{Otherwise} \end{cases}$$
$$\forall e_i^{\mathcal{C}1} \in \mathcal{E}_t^{\mathcal{C}1}, \forall e_i^{\mathcal{C}2} \in \mathcal{E}_t^{\mathcal{C}2}$$

Finally, The set vertices, $\mathcal{V}$, introduced in Sec. 2.2.3 is defined as:

$$\mathcal{V} = \{\mathcal{O},\ \mathcal{S},\ \mathcal{R}\} \tag{2.10}$$

Where $\mathcal{O}$, $\mathcal{S}$, and $\mathcal{R}$ were previously defined as the set of encounters, the set of scheduled nodes and the set of persistent nodes, respectively, with the condition $\mathcal{O} \cap \mathcal{S} \cap \mathcal{R} = \emptyset$. Finally, from Equations (2.10) and (2.9), the definition of the temporal graph is:

$$G_t = (\mathcal{V},\ \mathcal{E}_T)$$
$$s.t.$$
$$\mathcal{V} = \{\mathcal{O},\ \mathcal{S},\ \mathcal{R}\},$$
$$\mathcal{E}_T = \lambda\left(\mathcal{E}_t^{\mathcal{C}1},\ \mathcal{E}_t^{\mathcal{C}2}\right) \bigcup \mathcal{E}_t^{\mathcal{C}3} \bigcup \mathcal{E}_t^{\mathcal{C}4}$$

2.3 Conclusion

In many aspects, today's networks are increasingly heterogeneous. They require architectures and solutions that foster heterogeneity and render it seamless and advantageous. Heterogeneity manifests itself is various aspects discussed in this chapter and in Chapter 1. It can be apparent in the diversity of devices (i.e. their hardware and software aspects), communication media and protocols, quality requirements, traffic and channel types, mobility models and so on. Even though building an architecture that encapsulates all these heterogeneity aspects is challenging, achieving such an architecture is very rewarding. Our proposed contact classification model incorporates different aspects of heterogeneity, which makes it easy to extend for elaborate network models. It provides an in-depth understanding of the dynamics of such networks aiding in the design of efficient routing algorithms.

In this chapter, we emphasized the importance of fostering heterogeneity in designing current and future network architecture and provided a novel contact classification for heterogeneous wireless network, which captures heterogeneity aspects in a mathematical categorization of the network contacts. Through this classification, we can design elaborate network models fostering heterogeneity for future efficient architectures, which inherently support innovation in technology and application; and network scalability and robustness.

Chapter 3

Single-Attribute and Multi-Attribute Routing and Scheduling for Space Networks

3.1 Introduction

Space Networks referred to as the Interplanetary Network (IPN) [17] was designed to enable the communication between different deep space objects. Therefore, this network has for long been of great interest mainly to the government space agencies, which then represented the major player in deep space exploration and interplanetary science missions. With the advances in technology, new types of missions are envisaged for the near future and more companies in the private sector are showing interest in the deep space exploration. These missions are imagined differently with more audacious goals of establishing colonies of robots, and eventually humans, on other planets such as Mars and in the cis-lunar space [9, 18–21]. The National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL), for example, is planning the Mars 2020 mission [22]. In this mission, a rover and a drone called Mars Helicopter Scout (MHS) will be sent to the surface of Mars to investigate the habitability of Mars and the possibility of past life, among other goals. Another example of these missions is the recent Chinese Lunar Exploration Mission. Chang'e 4 [23]. It accomplished the

first successful soft landing on the far side of the Moon with rover Yutu-2 in addition to the insertion of the communication satellite, Queqiao. The success of this mission and the past ones are paving the way for more ambitious missions.

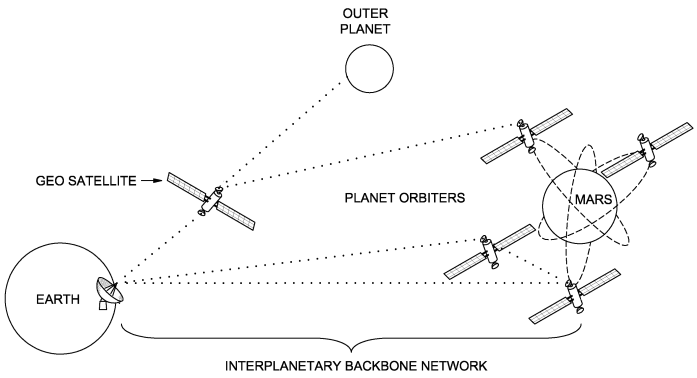


Figure 3.1: An illustration of the Space Network (source: [1] - Modified)

With changes in future space exploration missions, it is expected that IPNs network density, which has been relatively sparse, will significantly increase. Consequently, the types of network data will change to potentially include intra-colony communications, inter space objects messages, in addition to the usual payload sent to Earth-based mission operation centers (MOC) from the space objects. This new network traffic will have varying priorities and message sizes among other properties. *Scalable and robust routing algorithms* are, hence, needed to facilitate communication in current IPNs and as it evolves with future missions.

Among the existing IPN routing algorithms, the contact Graph Routing (CGR) [24, 25] is the most prominent one. CGR was proposed to overcome the main characteristics that are found in DTNs and not in Earth-bound network: four main features of DTN: (1) lengthy round-trip times of communication, (2) the recurrent absence of relay nodes and communication windows, (3) the absence of end-to-end paths, (4) the asymmetric data rates. CGR was then integrated with the Bundle Protocol (BP). The BP, which will be introduced in more details in Section 3.2, was approved by the Consultative Committee for Space Data Systems

(CCSDS) in September 2015 and published as a Blue book becoming the recommended protocol for DTN-based IPNs [26]. CGR has been proved to efficiently route data in the IPN [27], nevertheless, and as we show in the results section, CGR only performs well in space networks with a relatively small contact plan (i.e. the schedule of contacts between every pair of nodes). Finally, the currently used and proposed routing algorithms, mainly [24], and [28], follow a greedy approach of finding the best feasible path for each message (a.k.a. bundle described in Section 3.2) at a time ignoring the overall usage of the network. While this increases the single bundle optimality (in terms of earliest arrival time), it does not necessarily guarantee an optimal utilization of the overall network.

While the IPN has particular routing challenges, it also presents the problem of a severe limitation of resources (such as link connectivity, link bandwidth, processing capacity of nodes etc.) As mentioned earlier, DTNs are characterized by the intermittent connections between nodes; therefore, it is important to implement a routing algorithm that takes this issue into consideration. *Message scheduling* has been one of the solutions to the problem of optimal resource utilization; however, it has not been explored in the context of routing in IPNs as we detail in Section 3.2. Routing protocols in IPN assume the availability of contact plans (i.e. the list of contact windows between relevant pairs of nodes in the network) in order to find the best temporal path (i.e. a sequence of temporal contacts) for every message. In a sparse network, the traffic load is not very significant permitting the use of the greedy approach to route each message at a time. However, as the traffic load and types increase, it will be important to schedule messages for routing in order to ensure a more optimal use of the contacts.

As mentioned earlier, the existing routing protocols are either not optimized for the IPN or just not for the aforementioned new challenges. We propose N-

Look Ahead Routing and Scheduling (N-LARS) algorithms [5]. Using N-LARS, each node in the network sorts the outgoing buffer based a certain criteria before it routes the message and sends it to the next hop. This scheduling and routing algorithm has shown positive results, but it was limited by sorting criteria. Consequently, we introduce a novel solution that uses Multi-Attribute Decision Making coupled with both scheduling and routing specially designed for the IPN architecture. Our contributions are as follows. We propose the first mathematical model for the problem of scheduling and routing in scheduled networks (i.e. networks where the schedule of nodes' encounters is pre-loaded in each node as part of the mission planning and operation process) and, in particular, deep space networks. We then propose MARS (Multi-Attribute Routing and Scheduling), a Multi-Attribute Decision Making (MADM) based look-ahead scheduling algorithm with a sliding window of width n.

Through this scheduling and routing algorithm, we study the effect of adopting an MADM technique in the message scheduling process and show how that affects the performance of the network. Our study shows that using larger window sizes for scheduling increases the number of messages delivered to their final destinations. It further shows that considering multiple criteria to choose the next best message has a direct effect on the performance. We also compare the performance of our algorithm to that of CGR using large-scale and small-scale network simulations. The results show that our proposed MARS algorithm delivers up to 3 times more bundles than CGR in large-scale networks and 5.7% more bundles in smaller-scale networks. It also significantly reduces the overhead and the average end-to-end delay.

The chapter is organized as follows. We start by providing background and related work in Section 3.2. We then introduce the formal definition of the schedul-

ing and routing problem in Section 3.3. In Section 3.4, we introduce N-LARS along with its evaluation. Then in Section 3.5, we present MARS, the scheduling and routing algorithm along with the MADM technique that we chose to use. The experimental setup and numerical results are detailed in Section 3.6. Finally, we conclude our work and provide future directions in Section 3.7.

3.2 Background and Related Work

In this section, we give a brief introduction to IPN. We list the works that have been done on routing and scheduling in the IPN. We introduce MADM and describe some of its known techniques.

3.2.1 Space Networks

The Interplanetary Internet is a network composed of interconnected space objects, which are in turn connected to mission control stations on the surface of Earth. this network is our ears and eyes in the deep space. It has been a very sparse network with very few space shuttles until recently. In fact, space mission are experiencing a great increase; NASA's Mars 2038 [29], for example, is planned with human presence on Mars. This entails many changes to the IPN and its architecture, and hence the communication strategies used.

3.2.1.1 Routing in Space Network

IPN is inherently delay tolerant because of the lengthy round-trip times of communication, the recurrent absence of relay nodes and communication windows, the absence of end-to-end paths, the asymmetric data rates and the high error rates. These characteristics make the Internet communication protocols inappli-

cable to IPN. DTN is built on the store-carry-forward principle because each node has the capability of storing a message and being its custodian until it finds the next hop that the message can be forwarded to. Each node, however, is able to decline custody of a message for a certain destination, in which case, it is excluded from the route computation for messages to that destination until the node accepts custody [12, 30].

In order to define and shape the use of DTN in IPNs, the Bundle Protocol (BP) was proposed in RFC 5050 [30] and later approved by the CCSDS as a recommended standard that "defines end-to-end protocol, block formats, and abstract service descriptions for the exchange of messages (bundles) that support Delay Tolerant Networking (DTN)." [26] As such, the BP dictates different fields in a data unit called "bundle," the role of each node in the network referred to as "bundle nodes," the bundle processing including forwarding it from a node to another, in addition to other administrative and architectural implementation details. In fact, it provides the baseline requirement for forwarding and the role and responsibility of bundle nodes in the process; nevertheless, the BP does not include any specific routing technique making it more flexible and applicable to specific network design and environmental requirements.

Several works have ben proposed for routing in the IPN [6, 28, 31], but we only use CGR in this work since the most prominent one. The Contact Graph Routing (CGR) was then proposed in an IETF draft [24, 25, 32] by Scott Burleigh at NASA JPL. It has then been improved through various efforts such as the work in [33], which incorporated the use of standard Dijkstra's algorithm, another effort that added the use temporal route lists in order to improve the execution time [34], and one by S. Burleigh et al. himself where CGR was extended to be used in opportunistic networks in addition to the scheduled networks it was

defined for [35], and yet another work that included bundle fragmentation in an effort to enhance network performance [36]. CGR with the later enhacement is being standardized by the CCSDS as part of the draft recommended standard Schedule-Aware Bandle (SABR) Red Book [37]. A thorough description of CGR and its development has been published in [27].

3.2.1.2 Scheduling in Space Networks

As for scheduling in DTN-based IPN, our literature review showed that scheduling has not been studied nor has it been applied to deep space networks and/or scheduled networks. Scheduling, together with routing has been more thoroughly studied in wireless networks, optical networks and many other types of Earth-bound networks then in DTNs. An early application of scheduling and routing to DTNs by Wolf et al. [38], in which they have used Genetic Algorithms to assign tasks to a limited number of Earth observing satellites (EOS), has been proved to enhance their performance. However, this work is only applied to task assignment and not to routing. In [39], Xian et al. proposed a look ahead routing algorithm for Mobile ad-hoc DTN networks. Their Ferry-Gateway based approach uses the look ahead information about the future encounters of gateways with ferries in order to anticipate the routing of a bundle to the best choice ferry enhancing the QoS of the network. This technique depends on the ability of the gateway nodes to foresee the arrival of a ferry and choose the best among them. The application of scheduling techniques to the outgoing message queue could similarly enhance IPN's performance and that is what we study in this chapter.

3.2.2 Multi-Attribute Decision Making (MADM) Optimization

The pursuit of an ideal solution given several criteria has been the incentive to design new techniques that allow decision makers to select the best options. Multi-Attribute Decision Making (MADM) [40] was introduced to enable informed decision-making, which is based on the set of available options (alternatives) and their attributes that affect their performance if chosen. MADM has been studied for several decades resulting in a multitude of techniques that differ in the type of data they are optimized for, and the optimization issue that is to be solved. Some of the most widely used and studied MADM methods that we consider using for this work are: Vise Kriterijumska Optimiyacija I Kompromisno Resenje (Multicriteria Optimization and Compromise Solution – VIKOR) [41], Preference Ranking Organization Method for Enriched Evaluation (PROMETHEE) [42], Technique of Order Preference Similarity to the Ideal Solution (TOPSIS) [40, 43], and the simplest and most intuitive, Simple Additive Weighting (SAW) [40].

Sabaei et al. summarized the types of MADM methods in [44] and provided a guide on how to choose the best type for a problem. In fact there are methods which are designed for problems where criteria and attributes are not available and those in which they are. The latter is further classified by differentiating between methods that perform a ranking of the alternatives and those that provide pair-wise comparisons. More importantly, there are methods that require more interaction with the user than others [45]. MADM has been applied to Interplanetary Networks in [46], where Bisio et al. studied the effect of MADM techniques in mitigating congestion when used to select the next hop. In their study, they showed that TOPSIS outperformed the other methods used in the experiments. MADM, however, was not used for message scheduling and routing in the net-

work.

3.3 Formal Definition of the Scheduling and Routing Problem in Space Networks

In this section, we provide the mathematical formulation of scheduled DTN contacts and of the MARS algorithm. To model the scheduling and routing problem, we start by creating the model using a single attribute for scheduling, then we augment the problem to use multiple attribute for the scheduling of bundles. We first present the general network notations in Section 3.3.1 and then we introduce the details of the single-attribute model corresponding the N-LARS in Section 3.3.2 followed by the multi-attribute scheduling model in Section 3.3.3, which is then implemented using the MARS heuristic.

3.3.1 General Notation

The network will be represented as a temporal graph that we denote by G_t. The temporal graph is characterized by a set of temporal edges, $\mathcal{E}_t$, where an edge between nodes u and v has multiple occurrences with different start times and duration. Hence we write $G_t = (\mathcal{V},\ \mathcal{E}_t)$. The contacts (i.e. resources) are intermittent in this environment. Network edges are hence discontinuous and occur at different times with, usually, a different duration in each occurrence. For every pair of adjacent nodes u and v, we write $\mathcal{E}_{t,(u,v)} = \{e_{t_1}, e_{t_2}, ... | e_{t_i} \in \mathcal{E}_t\}$ to denote the set of all temporal edges between u and v, $\forall u, v \in \mathcal{V}$, $s.t.\ u \neq v$. Furthermore, we define an edge follows: $e_{t_i} = (t_s, \Delta t, r_{up}, r_{down})$

The variable t_s is the start time of the edge, Δt is its duration, and r_{up} and r_{down} are its uplink data-rate and downlink data-rate respectively. Some of the nodes,

however, could be constantly connected; therefore, there is only one continuous edge $e_{(a,b)}$ between them such that $e_{(a,b)} \in \mathcal{E}_t$. In such a case, a continuous edge $e_{(a,b)}$ can be considered as a special case of $e_{(a,b),t}$ where $t_s = t_\emptyset$ ($t_\emptyset$ is the reference time), $\Delta t = \infty$, and $r_{up} = r_{down}$ in case it is symmetric.

3.3.2 Single-Attribute Scheduling and Routing Model

Table 3.1: Formulation parameters

Parameter	Description
$\widehat{\mathcal{Q}}$	Set of requests (i.e. bundles)
q	Request q, referred to as bundle in the context of BP
q^s	Size of request/bundle q
q^t	Preferred start time of request q
q^{TTL}	Time to live of request/bundle q
q^p	Priority of request/bundle q
q^{src}	Source node of request/bundle q
q^{dst}	Destination node of request/bundle q
$\mathcal{A}$	Set of the requests' attributes of size m (i.e. ttl, size, ...)
w_a	Weight of attribute a s.t. $a \in \mathcal{A}$
$\mathcal{E}_t$	Set of all temporal edges
$\mathcal{E}_{t,(u,v)}$	Set of all temporal edges between nodes u and v
e^t	Start time of edge e
$e^t_{r,q}$	Arrival time of request q to temporal edge e
$e^{\Delta t}$	Duration of temporal edge e
e^r	Datarate of temporal edge e
e^{src}	Source node of temporal edge e
e^{dst}	Destination node of temporal edge e
ε_e	Accounts for the time spent in the queue (at e^{src}) and other delay variants on edge e
ε^*_e	The part of ε_e that is independent of the bundle such as the one way light time (OWLT)
n	Number of requests considered for MARS

In this routing-scheduling problem there are two main goals that need to be

achieved. First, finding a complete feasible temporal path for a set of bundles that is globally optimal; i.e. maximizing the utilization of the overall network by delivering a higher number of bundles as apposed to greedily delivering individual messages using their earliest arrival path; and the second transmitting higher priority messages before lower priority ones. When solving this problem for a single demand with no knowledge of the state of the network, the solution is straightforward. However, when we consider a window of n messages to be scheduled for transmission, the problem can be formulated as follows.

Table 3.2: Formulation variables

Variable	Description
$x_{q,e}$	Equals 1 when edge e is in the path for request q
b_q	Equals 1 if the request q is scheduled for transmission
$\overline{b_q}$	$1 - b_q$

We first define the set of all requests (i.e. bundles) $\widehat{\mathcal{Q}}$ and denote each element as q. Every request q has a preferred start time q^t and a Time To Live (TTL) q^{TTL}, which defines the time when it will be dropped if it does not reach its final destination. It also has a total size q^s and a priority q^p. The set $\widehat{\mathcal{Q}}$ can be divided into two subsets $\overline{\mathcal{Q}}$ and $\mathcal{Q}$. The former is the set of all requests that could not be satisfied and the latter is those with a feasible path to the final destination. The condition $\overline{\mathcal{Q}} \cap \mathcal{Q} = \emptyset$ must hold under this definition.

The following is our formulation of the N-LARS algorithm. The parameters and variables used in the model are listed in Tables 3.1 and 3.2 respectively.

$$min \sum_{q \in \widehat{\mathcal{Q}}} \sum_{e \in \mathcal{E}_t} ((max(e^t, e^t_{r,q}) + \frac{q^s}{e^r} + \varepsilon_e) \times x_{q,e}) + \sum_{q \in \widehat{\mathcal{Q}}} (C^{q_i^p} \times \overline{b_q})$$

Subject to:

$$\sum_{q \in \widehat{\mathcal{Q}}} (\frac{q^s}{e^r} + \varepsilon_e^*) \times x_{q,e} \leq e^t + e^{\Delta t}, \forall e \in \mathcal{E}_t \tag{3.1}$$

Constraint (3.1) ensure that the duration of all the requests routed through edge e does not exceed the temporal availability ($e^t + e^{\Delta t}$) of the edge e.

$$x_{q,e} \times e^t \leq q^{TTL} + q^t, \ \forall e \in \mathcal{E}_t, \forall q \in \widehat{\mathcal{Q}} \tag{3.2}$$

Constraint (3.2) ensures that an edge e is used for request q only if it starts before the request expires at time $t = q^{TTL} + q^t$.

$$x_{q,e} \times q^t \leq e^t + e^{\Delta t}, \ \forall e \in \mathcal{E}_t, \forall q \in \widehat{\mathcal{Q}} \tag{3.3}$$

Constraint (3.3) guarantees that the edge does not end (time $e^t + e^{\Delta t}$) before the request is issued.

$$\sum_{q \in \widehat{\mathcal{Q}}} b_q - \sum_{q \in \widehat{\mathcal{Q}}} \sum_{e \in \mathcal{E}_{t,(q^{src},x)}} x_{q,e} = 0, \ \forall x \in \mathcal{V} \tag{3.4}$$

Constraint (3.4) maintains the flow conservation at the source node ensuring that the number of satisfied requests ($\sum_{q \in \widehat{\mathcal{Q}}} b_q$) is equal to the number of temporal outgoing edges of this node that were used in the final schedule ($\sum_{q \in \widehat{\mathcal{Q}}} \sum_{e \in \mathcal{E}_{t,(q^{src},x)}} x_{q,e}$).

$$\sum_{q \in \widehat{\mathcal{Q}}} b_q - \sum_{q \in \widehat{\mathcal{Q}}} \sum_{e \in \mathcal{E}_{t,(x,q^{dst})}} x_{q,e} = 0, \ \forall x \in \mathcal{V} \tag{3.5}$$

Constraint (3.5) does the same thing as Constraint (3.4) at the destination nodes.

$$\begin{aligned} \sum_{e \in \mathcal{E}_{t,(v,u)}} x_{q,e} - \sum_{e \in \mathcal{E}_{t,(u,z)}} x_{q,e} = 0, \ \forall q \in \widehat{\mathcal{Q}}, \\ \forall u, v, z \in \mathcal{V}, \ u \neq v \neq z \end{aligned} \tag{3.6}$$

Constraint (3.6) ensures flow conservation at all intermediate nodes. For every request traversing this node, the number of incoming edges used for the request must equal the number of outgoing edges used for that same request. Note: the two sums $\sum_{e \in \mathcal{E}_{t,(v,u)}} x_{q,e}$ and $\sum_{e \in \mathcal{E}_{t,(u,z)}} x_{q,e}$ are at most 1 (see Constraint (3.7)).

$$\sum_{e \in \mathcal{E}_{t,(u,v)}} x_{q,e} \leq 1, \ \forall q \in \widehat{\mathcal{Q}} \, \forall u, v \in \mathcal{V}, \ u \neq v \tag{3.7}$$

Constraint (3.7) is used to make sure that among all the temporal edges between u and v ($s.t.\ u, v \in \mathcal{V}, u \neq v$) at most one is chosen for each request q.

$$b_q \geq x_{q,e}, \ \forall q \in \widehat{\mathcal{Q}}, \forall e \in \mathcal{E}_t \tag{3.8}$$

Constraint (3.8) is used to indicate whether a message is scheduled for transmission or not.

$$\begin{gathered} x_{q,e_1} \times e_1^t < e_2^t + e_2^{\Delta t}, \ \forall q \in \widehat{\mathcal{Q}}, \\ \forall e_1 \in \mathcal{E}_{t,(v,u)}, \ \forall e_2 \in \mathcal{E}_{t,(u,z)}, \\ \forall u, v, z \in \mathcal{V}, \ u \neq v \neq z \end{gathered} \tag{3.9}$$

Constraint (3.9) is used to enforce the temporal precedence of edges in the same route. If edge e_1 is chosen for request q; then, all subsequent edges e_2 satisfying $e_1^{dst} = e_2^{src}$ must overlap with e_1.

$$x_{q,e} \in \{0, 1\}, \ \forall q \in \widehat{\mathcal{Q}}, \forall e \in \mathcal{E}_t \tag{3.10}$$

$$b_q \in \{0, 1\}, \ \forall q \in \widehat{\mathcal{Q}} \tag{3.11}$$

Constraints (3.10) and (3.11) make x_i and b_q binary variables, respectively.

$$\sum_{q \in \widehat{\mathcal{Q}}} b_q <= n \tag{3.12}$$

Constraint (3.12) ensures that at most n requests are satisfied (i.e. no request is sent twice).

$$n \in \mathbb{Z}^+ \tag{3.13}$$

In constraint (3.13), n is a finite number, and this ensures that n is a positive integer that is greater than or equal to one.

Our objective function aims at minimizing the overall delay of the n transmitted bundles. It is composed of two main parts. The first part ensures that the paths chosen for each one of the requests that can be satisfied are earliest arrival paths. This summation of the time it takes a bundle to traverse an edge e if chosen (i.e. $x_{q,e} = 1$) does **not** ensure each particular path ensure the earliest arrival time of a bundle, but that **all paths constitute an optimal schedule** that schedules and delivers the highest number of bundles. That is to say, some paths could be k^{th} shortest path in order for more bundles to be routed, while making sure the overall delay is minimal. The second part of the summation ensures the requests with higher priorities are routed before the lower priority ones. C is large constant in time units and is raised to a power equal to the priority of the request. The resulting number is multiplied by the complement of b_q. The latter ensures that we add the penalty for the unsatisfied requests only, and the former is to make the objective function yield a value much higher when the lower priority bundles are preferred over higher priority ones. Hence, this second part adds a penalty to the delay reflecting the priority of the declined request. Note that the higher the

value of q^p, the higher its priority.

3.3.3 Multi-Attribute Scheduling and Routing Model

Table 3.3: Extra Formulation parameters

Parameter	Description
$\mathcal{A}$	Set of the requests' attributes of size m (i.e. ttl, size, ...)
w_a	Weight of attribute a s.t. $a \in \mathcal{A}$

As we introduce the multi-attribute decision making approach to the model, the problem formulation has to slightly change in order to reflect for the new changes. Hence, the objective function is defined as follows. The parameters used in the model are listed both in Table 3.1 and Table 3.3, and the variables are listed in Table 3.2.

$$min \sum_{q \in \widehat{\mathcal{Q}}} \sum_{e \in \mathcal{E}_t} ((max(e^t, e^t_{r,q}) + \frac{q^s}{e^r} + \varepsilon_e) \times x_{q,e}) + \sum_{q \in \widehat{\mathcal{Q}}} ((\sum_{a \in \mathcal{A}} C^{q^a_i} \times w_a) \times \overline{b_q})$$

Subject to:

Constraints (3.1)-(3.13), and

$$\sum_{a \in \mathcal{A}} w_a = 1 \tag{3.14}$$

Constraint (3.14) ensures that the sum of all attributes' weights is equal to one.

Our objective function still aims at minimizing the overall delay of the transmitted messages. It is also composed of two main parts. The first part is similar to the one used in the Equation (3.1). The second part of the summation incorporates the MADM aspect into the formulation. it ensures the requests with higher value of their attribute are routed before the ones with a lower overall value. C is a large constant in time units and is raised to a power equal to the value of each attribute of the a request and then multiplied by the weight of the attribute. The

resulting number is multiplied by the complement of b_q. The latter ensures that we add the penalty for the unsatisfied requests only, and the former is to make the objective function yield a value much higher when the lower value messages are preferred over higher value ones. Hence, this second part adds a penalty to the delay reflecting the attributes and weights of the declined request.

3.4 N-Look Ahead Routing and Scheduling Algorithm (N-LARS)

3.4.1 N-LARS Heuristic

The model described in Sec. 3.3 provides a detailed and precise representation of the scheduled DTN deep space networks; however, this model is not scalable and cannot be run on a machine with limited resources. Based on the aforementioned definitions, we derive a decentralized algorithm that optimizes the number of bundles transmitted by compromising the local optima. That is to say, it does not transmit each bundle over its earliest arrival path (local optimum), but sends more bundles by scheduling n bundles at a time.The implementation of N-LARS heuristic has three main building blocks: (1) routing heuristic, (2) message transmission module and (3) reroute method.

Algorithm 1 starts by initializing the list *toBeRouted* with the first n elements in the list of all messages. It sorts them based on the heuristic criterion β, in lines 2–4. β could be the size of the bundles, their TTL, their priority and or a mixture of some or all of these. In lines 5–15, for each message in the sorted list, the algorithm checks whether the message already has a source route. If it does (line 6), the bundle is added the outgoing messages list; otherwise, the algorithm tries to find the shortest path (line 9) to the final destination using $findRoute$ method, which we adopt from out work in [6] and describe int he next paragraph. In

Algorithm 1 N-Look Ahead Routing and Scheduling Algorithm

1: **Input:** Temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$ in its edge stream representation, window size n, list of outgoing messages $msgList = m_1, m_2, ..., m_l$, the heuristic criteria β
2: $toBeRouted \leftarrow \{m_1, ..., m_n\}, m_i \in msgList$;
3: $sort(toBeRouted, \beta)$;
4: $outgoingMsgs \leftarrow \{\}$;
5: **for all** $msg \in toBeRouted$ **do**
6: **if** $msg.sourceRoute \neq NULL$ **then**
7: $outgoingMsgs \leftarrow outgoingMsgs \cup \{msg\}$;
8: **else**
9: $r_{msg} = findRoute(msg, G_t)$;
10: **if** $r_{msg} \neq NULL$ **then**
11: $msg.sourceRoute \leftarrow r_{msg}$;
12: $outgoingMsgs \leftarrow outgoingMsgs \cup \{msg\}$;
13: $UpdateG_t$ *egde utilization*;
14: **else**
15: $msgList \leftarrow msgList \setminus \{msg\}$;
16: **for all** $msg \in outgoingMsgs$ **do**
17: **if** $TryMsgToNextHop(msg) == failure$ **then**
18: $outgoingMsgs \leftarrow outgoingMsgs \setminus \{msg\}$;
19: $msg.sourceRoute \leftarrow NULL$;
20: $Reroute(msg)$;
21: **else**
22: $msgList \leftarrow msgList \setminus \{msg\}$;

lines 10–13, the path computed is then stored in the source route of the message, *msg.sourceRoute*. The capacity of each temporal edge on this path is then updated to reflect the fact that it will be transmitting this bundle. This step allows us to have a local rough estimation of the network utilization. If no path is found for this bundle across all the temporal edges of the network, it is dropped in line 15.

The algorithm $findRoute$ is an adaptation of the Dijkstra search to the MTG model introduced in [6]. The path construction follows the same major steps of Dijkstra algorithm: it retrieves the set of outgoing edges of the current node, picks the next best (the edges with the earliest arrival time, chose this edge and

starts over from the next node. However, it differs in the relaxation function. In a temporal graph, the edge can only be used if it overlaps with the lifetime of the messages to be routed (the edge does not end before the creation of the message and does not start after the expiration of the message's TTL) and if its data rate is enough to transmit the message from one end to the other within the edge's duration and the message's lifespan. Hence, in our routing algorithm, we eliminate the outgoing edges that do not meet these criteria in order to reduce the time complexity of the algorithm and ensure its correctness.

The second and third stages of the algorithm are interdependent and are performed in lines 16–22. In the second stage, the algorithm attempts to send a message to the next hop when the right temporal edge is up. Upon success, the message is removed from the message list (line 22). If the transmission fails, through lines 17–20, the bundle goes to the third stage, where its source route is set to $NULL$, and it is put back in the list of all message $msgList$ to be rerouted. In this step, we ensure that the bundles with a source route computed in other nodes will be corrected by the current custodian of the bundle. With this option, the algorithm adopts a hybrid of source and per-hop routing. It mitigates the drawbacks of lacking global network utilization knowledge.

3.4.2 Validation and Numerical Results

We implemented the single-attribute model in IBM ILOG CPLEX Optimization Studio [47] to generate the routing schedule for a set of requests and randomly generated networks. We run it on a 2 CPU/16 core machine with 32GB of RAM. Using this implementation, we obtained the oracle sub-optimal scheduling of bundles in the small network described in the following section. The CPLEX solver scheduled up to 87 bundles representing a 33% sub-optimal solution lim-

ited by memory. This value is compared to the results obtained from the simulations. We also implemented N-LARS routing heuristic in the Opportunistic Network Environment (ONE) simulator. The experimental setup and numerical results are presented in this sections.

3.4.2.1 Experimental Setup and Network Configuration

In order to test and analyze our N-LARS heuristic for scheduling and routing, we use randomly generated networks (N) based on ESA's project METERON [9], depicted in Fig. 3.2. The random networks are composed of 3 main segments: 1) Mars-based network connecting all the rovers, sensors and robots 2) Mars-Earth communication segment composed of layers of relay satellites and 3) Earth-based network. Each random network N will be composed of N_r layers of relay satellites (e.g. the networks in Fig. 3.2 has $N_r = 2$). In each network, a random number of nodes is generated for each layer (N_r) and for each of the network segments 1) and 3). Finally, the edges have varying data rates that are derived from realistic values.

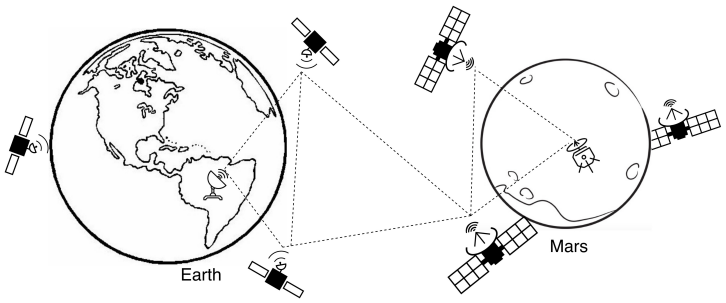


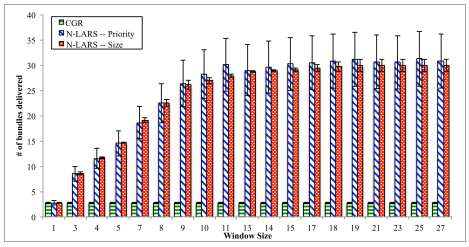
Figure 3.2: Illustration of a Mars-Earth small network used in the experiment.

The temporal edges duration are also derived form a realistic Mars-Earth contact plans generated using Webgeocalc Tool [48]. For each network setup, we generate a list of bundles and route them throughout the equivalent of 2 days. We vary the value of n and record the number of bundles delivered for each configuration. The buffer size also affects the routing performance depending on the

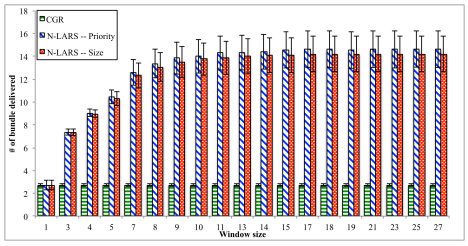
value chosen for n; hence, we run the same experiment with different buffer size configurations. We use two categories of bundles: large bundles with sizes between 50 KB and 80 KB, and small bundles with sizes between 1 KB and 8 KB. The bundles are linearly distributed over the 2-day period. Furthermore, each bundle is assigned a priority at the time of creation. We assign 5 priority with a highest priority equal to 1.

3.4.2.2 Numerical Results

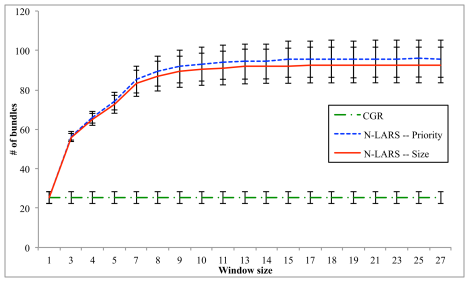
After running the experiment described above, we obtained the results that we analyze in this section. The generated networks have between 8 nodes with 344 temporal edges depicted in Figure 3.2 and 56 nodes with 11240 temporal edges. We ran 200 simulations in which we generate an average of 300 bundles to be transmitted from different Mars nodes to Earth nodes. We start by setting n to one which results in the same performance as CGR and show the results of changing the window size n in graphs in Figure 3.3. In our experiments, we did not study the effect of the TTL and limited our analysis to the priority and the size of the messages due to space limitations. Figure 3.3b shows the average number of bundles delivered across all simulations using different routing techniques. The graph shows that both N-LARS heuristics outperform CGR with larger values of n; they route up to 4 times more bundles. Hence, N-LARS schedules up to 16.09% of the sub-optimal results obtained using CPLEX solver, while CGR only routes 2.3% of it. The graph in Fig. 3.3a shows the same results for the large network experiments. The number of bundles delivered doubles since there are more temporal edges to construct routes. These results, show that for large networks, the optimal values for n is between 10-13 and that for smaller networks, the steady state is reached for values between 8-11.



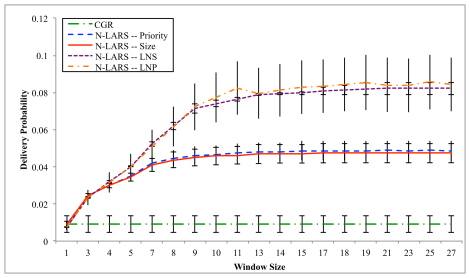
(a) The average number of bundles delivered using different routing techniques (56-node network).



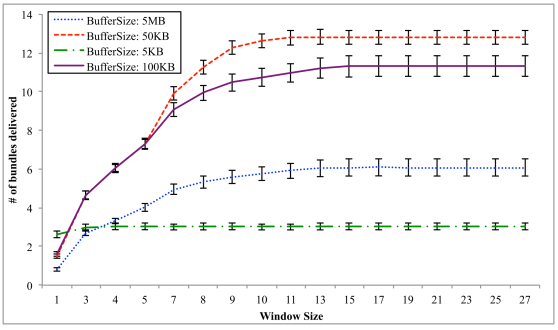
(b) The average number of bundles delivered using different routing techniques (8-node network).



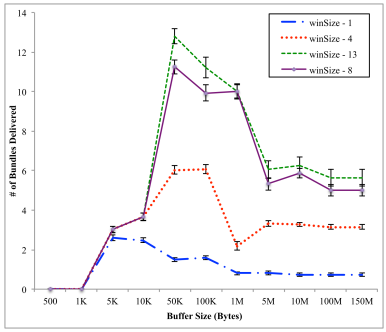
(c) The average number of bundles started but not delivered.



(d) Delivery probability with different window sizes and routing techniques (LNS: Large Network using Size, LNP: Large Network using Priority).



(e) The average number of bundles delivered under different buffer sizes and values of n.



(f) Effect of buffers sizes on the average number of bundles delivered.

Figure 3.3: N-LARS heuristic vs. CGR experimental results.

Figure 3.3c shows the number of messages attempted. Because of the lack of global knowledge of the network, most of the messages started are not delivered to the final destination. The rerouting of the bundles at intermediate nodes yields no new routes since it has a better knowledge of the network; Therefore, the

number of started messages is higher than the number of delivered messages. Fig. 3.3d shows the delivery probability that increases as we increase the window size. For the same ranges of n, the probability of delivery also reaches the steady state and, as expected, it is higher in larger networks. Moreover, the delivery probability is slightly better for smaller values of n with $\beta =$*"Size"*. After reaching the steady state, the heuristic with $\beta =$*"Priority"* performs better. This is due to the fact that there are only five possible priorities compared to the ten different sizes assigned to the bundles. We also note that there is a large fluctuation in these results as the error bars show in both Fig. 3.3b and Fig. 3.3a. This means that there was a larger fluctuation in the performance of N-LARS based on priority across the simulation, on the opposite of N-LARS based on size.

We also studied the effect of buffer sizes on the performance of our N-LARS heuristic. We show results from the 8-node network in the following. Fig. 3.3f shows that as the size of the buffer becomes larger than an optimal value, it starts having the reverse effect on the messages delivery. The optimal size of the buffer changes according to the value of n; therefore, the choice of n also depends on the buffer size. That is due to the fact that the messages that could have been routed, are pushed back to the end of the queue. They hence, have less chances to be reached by the router. Nonetheless, in general, the buffer size 50 KB yields the best performance in terms of the number of delivered bundles. The graph in Fig. 3.3e shows across all buffer sizes, the steady state occurs at the same range of window size values; i.e. between 8-11 for the small network.

3.5 The Multi-Attribute Routing and Scheduling Algorithm

The MILP formulation presented in the previous section is not practical for real-time scheduling and routing of messages because it does not converge in a reasonable amount of time for nodes that do not have large computation power. Therefore, we introduce the following MARS algorithm that is built on the constraints and objective of this definition, but that can be run efficiently. The proposed scheduling and routing algorithm, described in detail in this section, is a multi-attribute decision making based algorithm, which takes into consideration multiple attributes in the choice of the next n messages to schedule.

3.5.1 MADM for IPN Routing

A Multi-Attribute Decision Making process consists of three major steps: 1) deciding which attributes have a higher impact on the objective function, 2) determining the weights of these attributes considering the application, network, and so on, and 3) choosing the MADM method that is most suitable for the problem to be solved. In the following, we introduce the attributes that we consider for the MARS algorithm and provide the details of the weight computation along with the MADM method that we use.

3.5.1.1 Choice of Attributes and Computation of Their Weights

After evaluating the two heuristics: N-LARS with message size and N-LARS with message priority, which sort the n messages based on their size or priority, respectively, we conclude that they prioritize the bundles with small sizes or high priorities. Accordingly, we improved the model to base the decision on multiple attributes of a message by taking into consideration the main attributes of

a message: Time To Live (TTL), size and priority. To further maintain fairness, the scheduling algorithm must also first transmit the messages that have been in the buffer longer. Hence, we choose four attributes of the messages for the scheduling: 1) size, 2) priority, 3) TTL and 4) Time in the Buffer (TiB).

The computation of the weights should reflect the importance of these attributes as defined by different applications and communications. For instance, if the TTL is more important than the other attributes, then it is assigned a higher weight. In this work, we define two ways to assign the weights. The first option is Direct Weight Assignment (DWA) which consists of assigning an array of four elements that add up to 1 and which constitute the weights. The other way is by using the Analytic Hierarchy Process (AHP) [49] to compute the weights of the four attributes based on a 4×4 matrix of scores using the Eigenvalue method. AHP allows us to set a matrix of pairwise scores comparing the importance of each attribute to all the others, then by computing the Eigenvector based on this matrix, we find the aggregate weights of individual attributes. This resulting list of weights is then used by the MADM method described next.

3.5.1.2 PROMETHEE II Applied to Space Networks

In order to rank the messages in the sliding window of size n, we considered four of the most commonly used and studied MADM methods: SAW, VIKOR, TOPSIS and PROMETHEE. Simple Additive Weight (SAW) method is the simplest one as it scores every alternative by summing up the multiplication of each of its attributes by their respective weights. The best alternative is hence the one with the highest score. The main drawback of SAW is that the final score is significantly affected by the weights of the attributes, which reduces the accuracy of this method. VIKOR ranks the alternative by computing their closeness to the

ideal solution. The closer the alternative, the better it is. Similarly, in TOPSIS, the Euclidean distance between the alternative and the ideal solution (called the positive ideal solution) is computed, but also the Euclidean distance to the worst solution (a.k.a. the negative ideal solution). The best alternatives are therefore those closest to the positive ideal solution and the farthest from the negative ideal solution.

As we experimented with TOPSIS and VIKOR in our deep space network setups, these two methods had a shortcoming related to the nature of the ranking criteria. The messages to be ranked, sometimes, have attributes that are very close to each other in value; therefore, the distance between the alternatives is not very significant. This resulted in the two ranking methods returning values in their original order. This problem is mainly due to the fact that VIKOR and TOPSIS were initially designed for highly complex systems [50]. We finally chose PROMETHEE since it solves all the drawbacks of the other methods, and it is known to be simple and stable [42, 44]. PROMETHEE has six different ranking formats: (I) partial ranking, (II) complete ranking, (III) ranking based on intervals, (IV) continuous case, (V) net flows and integer linear programming, and (VI) representation of human brain. In this work we are interested in the PROMETHEE I and II and more specifically in II. While PROMETHEE I only returns the best options of all the alternatives, PROMETHEE II provides a sorted list of alternatives based on the criteria and their weights. Relevant details of the algorithm are included in Section 3.5.2; however, the reader can find the full formal definition in [51, 52].

3.5.2 MARS: the Multi-Attribute Routing and Scheduling Algorithm

The algorithm derived from the routing and scheduling model introduced in Section 3.3 is similar to N-LARS in the computation of the path using Algorithm 1; however, it is different in the choice of the messages to route next. Hence, we describe details of the MADM part of the algorithm and leave the details of the routing for the reader to check in [5].

PORMETHEE II compares every alternative to all the other alternatives based on their attributes and their respective weights and stores the sum. This comparison specifies the preference of reference alternative msg_r over the other alternative msg_o represented by function f in Equation 3.15 (defined in [51]).

$$\pi(msg_r, msg_o) = \sum_{c \in C} w_c f_c(msg_r, msg_o), \tag{3.15}$$

where C is the set of all the criteria, w_c is the weight of criterion c. In MARS, we compute the preference sum for an alternative over all the others in Step 1 of Algorithm 2 (lines 7–19), and this preference function is adopted from the work in [50]. In the definition of PROMETHEE, there are two parameters to be set beforehand to finalize the sorting process: (1) the indifference threshold, referred to as t_i, specifies the threshold at which there is an indifference between the two alternatives; (2) in contrast, the preference threshold, denoted by t_{sp}, is used to set a strict preference between two alternative. When the alternative msg_r is better than msg_o, these thresholds are used to check whether msg_o is strictly preferred to msg_r (i.e. $msg_r(c) - msg_o(c) <= t_{sp}$) or msg_r is preferred to msg_o (i.e. $msg_r(c) - msg_o(c) >= t_i$).

After computing the pairwise preference values of all the alternative/messages, we compute the leaving, incoming and global flows of all the messages as de-

Algorithm 2 MARS Algorithm

Input:
Temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$ in its edge stream representation
Window size n
List of all messages $msgList = \{m_1, m_2, ..., m_l\}$
List of criteria $C = \{c_1, c_2, ..., c_p\}$
Weights of the criteria $W = \{w_1, w_2, ..., w_p\}$

Step 1 – compute preference matrix:

$toBeRouted \leftarrow \{m_1, ..., m_n\},\ m_i \in msgList$ ▷ *toBeRouted* are the alternatives
for all $c \in C$ **do** ▷ compute f_c
  for all $msg_r \in toBeRouted$ **do**
    for all $msg_o \in toBeRouted\ and\ msg_o \neq msg_r$ **do**
      if $msg_r(c) <= msg_o(c)$ **then**
        $f_c(msg_r, msg_o) \leftarrow 0$ ▷ msg_o is preferred
      else if $msg_r(c) - msg_o(c) <= t_{sp}$ **then**
        $f_c(msg_r, msg_o) \leftarrow 0$
      else if $msg_r(c) - msg_o(c) >= t_i$ **then**
        $f_c(msg_r, msg_o) \leftarrow 1$ ▷ msg_r is preferred
      else
        $pVal = ((msg_r(c) - msg_o(c)) - t_{sp})/(t_i - t_{sp})$
        ▷ Compute a preference value, $0 < pVal < 1$
        $f_c(msg_r, msg_o) \leftarrow pVal$

Step 2 – compute in/out and global flows:

for all $msg \in toBeRouted$ **do**
  $in_{msg} \leftarrow \sum_{\forall m \in toBeRouted, m \neq msg} \pi(msg, m)$ ▷ π is defined in Eq. 3.15
  $out_{msg} \leftarrow \sum_{\forall m \in toBeRouted, m \neq msg} \pi(m, msg)$ ▷ π is defined in Eq. 3.15
  $global_{msg} \leftarrow in_{msg} - out_{msg}$

Step 3 – sort and route:

$sortedMsgs \leftarrow sortDescending(global);$
$Route - Transmit(sortedMsgs);$

scribed in the second step of the Algorithm 2 (lines 20–23). These flows are used by PROMETHEE to determine the outranking relation between different alternative. In the final step, we sort the alternatives based on their descending ordering of global flows and use that ranking for the routing method descried in the next section.

3.5.2.1 Route-Transmit Algorithm

Algorithm 3 Route-Transmit Algorithm

1: **Input:**
2: Temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$ in its edge stream representation.
3: List of n sorted messages *sortedMsgs*;
4: $toBeRouted \leftarrow sortedMsgs$;
5: $outgoingMsgs \leftarrow \{\}$;
6: Algorithm 1 lines 5–22

Algorithm 3 is as described in Section 3.4.1. The only difference is that the set of Message *toBeRouted* is the sorted list returned by Algorithm 2.

3.5.3 MARS Algorithm Proof and Complexity

3.5.3.1 MARS Algorithm Proof

We use three main criteria in this proof: the completeness (all possible cases are considered), correctness (any arbitrary case is processed properly) and termination (the algorithm terminates).

Completeness This algorithm designed such that the set of MADM criteria is set at network setup; therefore, it is assumed that the criteria are shared between all the messages that each node will receive. For example, in this work, we use size, priority, TTL and TiB, which are all guaranteed to be found in all messages.

Under this assumption, it is trivial that the algorithm is complete. That is, the only variable type processed by both Algorithms 2 and 1 is the type *Message*; consequently, under the assumptions that all these messages have the specified criteria, any message will be processed properly by the algorithms.

Correctness In order to prove the correctness of this algorithm we start by defining the correctness criteria. This algorithm is correct iff:

1. The preference value computed in *Step 1* of Algorithm 2 properly reflects the preference between two options/messages; i.e. for every two messages the function always returns the same preference value when using the same criteria.

2. The message preference returned by the PROMETHEE II is preserved throughout the execution of the algorithm.

3. The algorithm routes the chosen messages in the order of preference and in their earliest arrival paths.

We further note that the correctness of the used MADM method (performed in Steps 1 and 2 lines 7-23 of Algorithm 2) is inherent from the correctness of PROMETHEE II; however, we include a short proof for our implementation of Step 1 in the following.

In order to prove the first criteria, we use the illustration in Figure 3.4. There are four branches in *Step 1* of Algorithm 2. The first case is when the value of msg_c is less or equal to the value of the other message msg_o, then the latter is preferred to the former. This case is straightforward and is not included in the illustration. The following three cases are used when the value of msg_c for a certain criteria is greater than that of the other message msg_o. The two thresholds

defined in Section 3.5.2 along with the *diff* are used to decide about the preference value, where *diff*=$msg_c(c) - msg_o(c)$. The interval number *1* illustrated in Fig. 3.4 shows the first branch where the difference is not enough to decide that msg_c is referred to msg_o; therefor, msg_o is given preference. The interval number *3* shows the opposite where msg_c is preferred. Finally, interval number *2* assign the fraction a/b (both shown in Figure 3.4) to the preference value. Figure 3.4 shows, the preference value of the two messages is dependent on (1) their criteria values and (2) thresholds t_i and t_{sp}. Since t_i and t_{sp} are set at the start of the network and are not changed in our algorithm at any time, we conclude that, for every two messages and criterion, the preference value is preserved across various iterations. We also conclude that it is consistent throughout the lifetime of the network.

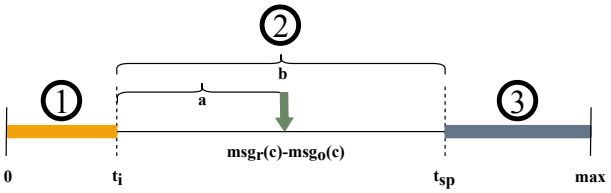


Figure 3.4: Preference value intervals.

The proof that the message ordering returned by PROMETHEE II is not changed throughout the algorithm can be found in the flow of the latter. In line 24, we obtain the sorted list of messages according to PROMETHEE's preference functions; therefore, we check the flow in Algorithm 1. The for-loop in line 5 transfers the messages one by one to a new list *outgoingMsgs* while following the order of the original list *sortedMsgs*. However, if a message cannot be router, it is not added to the list of outgoing messages and is also removed from the list of all messages, hence it does not affect the preference among all other messages that are to be routed. Finally, the messages are routed in their order of preference in the for-loop (line 16). In this loop, if two messages are to be transmitted over the same temporal edge, then they are sent according to their ordering in

outgoingMsgs, hence giving the resources to the first message in the list. As there are no instances in the algorithm where the order is changed, we conclude that the message preference returned by the PROMETHEE II is preserved throughout the execution.

Last but not least, Algorithm 1 uses the function $findRoute$ described in Sec 3.5.2.1. This function is a modified Dijkstra algorithm that uses a temporal graph model. Since the Dijkstra logic is preserved in the route construction, the correctness of this section follows from the correctness of Dijkstra algorithm. Therefore we conclude that the algorithm routes the chosen messages in their earliest arrival paths.

Termination At any given time period Δt, there is a finite number of messages in the buffer. There are three obvious reasons for this: (1) the buffer has a fixed and limited space, (2) the messages get dropped when their TTL expires and (3) the message creation rate in the whole network follows a certain functions which cannot be infinite since there is a finite number of nodes with limited hardware capabilities. As defined in Section 3.4, we note that the number of criteria used for sorting is finite: i.e. $|C| = p, p > 0$ as well as the window size, which is an integer greater than zero.

The two algorithms contain multiple for-loops to consider for termination.

Algorithm 2 line 8 : loops over all the criteria, which terminates since there is a finite number of the latter.

Algorithm 2 lines 9, 10 and 20 : are bounded by the number of messages in the *toBeRouted* set, and hence they are bound by the window size n. Since n is a finite number, the for-loops terminate.

Algorithm 1 line 15 : iterates over all the elements in the list *sortedMsgs* which is of size n or less. Therefore, this for-loop's always terminates.

Algorithm 1 line 22 : loops over *outgoingMsgs* that is composed of the messages from *sortedMsgs* with a route. Hence, $|outgoingMsgs| <= n$. This entails that this loop always terminates.

Since all the for-loops terminate, we conclude that both Algorithm 2 and Algorithm 1 terminate. ■

3.5.3.2 Complexity Analysis

Algorithm 2 can be split into two parts: (1) PROMETHEE II implementation until line 24 and (2) algorithm $route - Transmit$.

The first part heavily depends on n and it runs in:

$$C \times n \times n \times constant + 3n + log(n) \rightarrow Cn^2 + n + log(n) \tag{3.16}$$

Where C is the number of criteria, n is the window size and $log(n)$ is used for the sorting algorithm and depends on the sorting method chosen.

The second part runs Dijkstra algorithm in the input temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$; therefore, its worst case time complexity is:

$$\begin{aligned} n \times (|\mathcal{E}_t| + |\mathcal{V}| \times log(|\mathcal{V}|)) + constant \times n \\ \rightarrow n \times (|\mathcal{E}_t| + |\mathcal{V}| \times log(|\mathcal{V}|)) + n \end{aligned} \tag{3.17}$$

From Equations 3.16 and 3.17, the worst case time complexity of MARS is:

$$Cn^2 + n + log(n) + n \times (|\mathcal{E}_t| + |\mathcal{V}| \times log(|\mathcal{V}|))$$

because n and C are significantly smaller than the size of the graph, we could reduce the complexity to that of the function $findRoute$.

3.6 MARS Experimental Setup and Results

We used the Opportunistic Network Environment (ONE) simulator version 1.6.0 [53] to evaluate the proposed routing and scheduling algorithm. We compare the performance of our proposed algorithm, MARS, to CGR and N-LARS with priority using two experimental setups, small-scale Earth-Moon-Mars network and large networks. One of the assessment metrics used is the Overall Performance Metric (OPM) which gives a global view of the performance of the network as it is based on the delivery ratio, the delay and the overhead and is computed as follows:

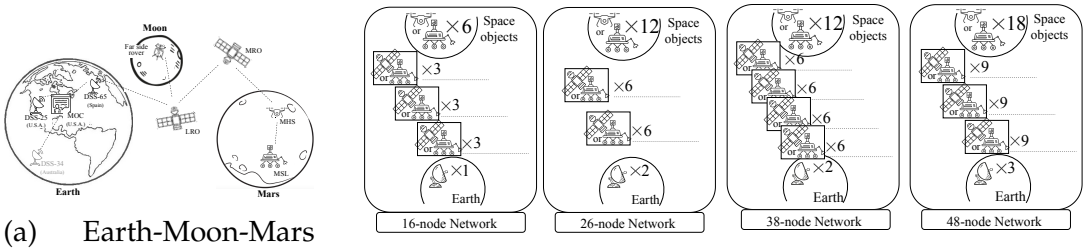
$$OPM = DeliveryRatio \times (1/AverageDelay) \times Goodput$$

where:
the delivery ratio=Delivered/Created,
bundle overhead ratio = (Relayed-Delivered)/ Delivered, and
message goodput=Delivered/Relayed [54].
The higher the value of OPM, the better the performance is. We present details of the implementation along with numerical results in this section.

3.6.1 CGR Implementation and Parameters

CGR was first implemented in the Interplanetary Overlay Networks (ION) [55], and then was introduced to the ONE simulator by Berlati et al. in [56] using ION version 3.6.0. The implementation provides two variations of CGR: 1) the contact Graph Routing in its original implementation designed for scheduled networks

only and 2) the Opportunistic CGR, which was enhanced to be used in networks that are not completely scheduled. We downloaded the "cgr-jni-Merge" package introduced in [56] and followed their steps to patch the ONE simulator (version 1.6.0) and merge in the new features provided by *cgr-jni-Merge*. In this work, we use deep space scheduled networks; therefore, we used the following CGR parameters. We used CGR as optimized for scheduled networks and implemented in the class *ContactGraphRouter.java*. We also used the new connections class provided by the *cgr-jni-Merge* package in order to set the speed of individual connections according to the contact plan. Finally, the implementation that we used, does not consider the priority while routing messages for the sake of performance comparison. We note that as of the time of submission, the latest available version of ION is 3.6.2, which was published in December 2018. Starting from version 3.6.1, CGR has been enhanced to align with the description of SABR that includes bundle fragmentation. It is however in the process of standardization and is still not stable.



(a) Earth-Moon-Mars networks simulation.

(b) Large networks setup examples.

Figure 3.5: Experimental network topologies.

3.6.2 Moon-Mars Small Network

3.6.2.1 Experimental Setup

In this experimental setup, we design and implement a network composed of objects from ongoing and future missions. As shown in Figure 3.5a, we use two nodes on Mars, Mars Science Laboratory (MSL) and Mars Helicopter Scout (MHS from NASA JPL's MARS 2020 mission [22]); one Mars orbiter, Mars Reconnaissance Orbiter (MRO); one lunar orbiter, Lunar Reconnaissance Orbiter (LRO); lunar rover Yutu-2 from Chang'e-4 mission on the far side of the Moon; and three Deep Space Stations (DSS) connected to the Mission Operation Center (MOC). The three DSS stations are located in the U.S.A., Spain and Australia to ensure constant coverage of the Earth. The choice of these objects is to demonstrate and simulate a diversity of objects and to use realistic networking specifications. While such a network is technically possible, it is currently not implemented.

We generated the contact plan for 24 hours using the Systems Tool Kit [57] and its AGI Standard Object Database. For convenience in experiment execution, we run the experiment by scaling down with a factor of 60 minutes and therefore run the simulation for 1440s. The data rates and traffic load used for this network are an adaptation from the work in [58]. The MOC sends telecommand bundles to all the satellites and rovers, which in return send acknowledgments and Telemetry data. Furthermore, all rovers and satellites send either one or two streams of science data to the MOC. The sizes of bundles are similar to the ones used in the same work, but we add a stream of science data composed of 1 MB bundles from MSL to the MOC. The frequency at which all these bundles are generated varies between 1 and 3 bunldes/minute per stream (the equivalent of 1-3 bundles/second in the scaled down experiment). The results are presented next.

3.6.2.2 Numerical Results

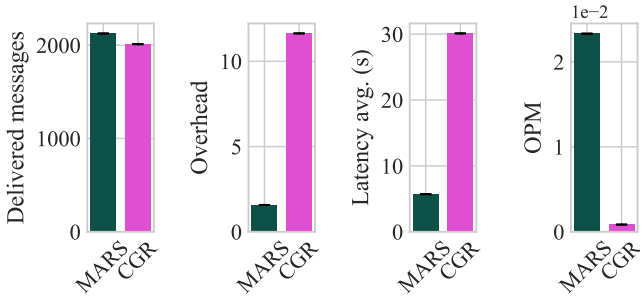


Figure 3.6: Earth-Moon-Mars network performance.

A total of 6196 bundles was generated, and as the results in Figure 3.6 depict, our proposed MARS algorithm increased the bundle delivery by 5.7% compared to CGR. It has also significantly improved the overhead and the average end-to-end delay. Finally, the OPM shows that, overall, MARS algorithm enhance the overall performance in this network setup. Nevertheless, the MADM aspect of the routing algorithm does not show any improvement by changing the window size, because there is not enough diversity in the traffic. In fact, the benefits of MADM will be greater and more noticeable in networks with higher traffic loads and diversity in bundle types, as we show in Section 3.6.3 presented next.

Table 3.4: Uplink and downlink data rates in Kbps

Node1	Node2	Data rates	Similar to
Rover	Satellite	32	MSL to Mars orbiting satellites [59]
Satellite	DSN stations	(13, 500)	MRO [60] Mars Odyssey [61] Mars Express [62]
Satllite	Satellite	128	Mars orbiting satellites [59]
MOC	DSN stations	1000	Assumed

3.6.3 Large-Scale Networks

3.6.3.1 ONE Network Setup

Table 3.5: Examples of network scenarios from real missions (The number of objects are not scaled because the missions are still limited in the deep space).

Network layout	Example
1-Layer	Mars rover (e.g. MSL) → (1) Mars Orbiting Satellite → Earth
2-layer	Mars rover (e.g. MSL) → (1) Mars Orbiting Satellite → (2) L2 Lagrange point halo orbiting satellite (such as Queqiao) → Earth
3-layer	Mars rover (e.g. Mars 2020 rover) → (1) Mars helicopter (e.g. Mars Helicopter Scout (MHS)) → (2) Mars Orbiting Satellite → (3) L2 Lagrange point halo orbiting satellite (such as Queqiao) → Earth
4-layer	Mars rover → (1) Mars Orbiting Satellite → (2) Moon rover (e.g. Yutu 2) → (3) Moon orbiting satellite → (4) GEO satellite → Earth

In order to evaluate the performance of MARS at different scales, we generate four networks two of which are depicted in Figure 3.5b. These network set-ups are not typically used currently but are not impossible to implement as the number of spacecrafts increases. Examples of such network with space objects from the current missions are provided in Table 3.5. These networks are composed of 16, 26, 38 and 48 nodes and composed of 4476 to 40824 temporal edges over a period of 3 days. We use scheduled networks of different layers of relay nodes between Earth and rovers on a different planet. These layers can be composed either of satellites orbiting Earth or other planets, such as Mars, or spacecrafts and rovers from various missions, such as Yutu-2 rover of Chang'e 4 [23]. The layers of relays have different sizes and are connected following randomly generated contact plans that mimic real-world deep space networks, such as the one

described in Sec. 3.6.2. The data rates of the links are also in ranges within the real-life data rates between different deep space satellites and between the latter and Earth-based stations as detailed in Table 3.4. For every network, we run three routing algorithms: MARS (with two different MADM configurations: Config 1 and Config 2), CGR, and N-LARS. In each run, we use 15 different traffic sets described in the next section.

3.6.3.2 Network Traffic

The traffic generated follows a Poisson arrival process with an arrival rate (λ) of 3, 9, and 15 bundles/min. The bundles are generated for a period of 3 days from a set of senders (i.e. nodes in the deep space) to a specific set of receiver nodes on Earth. The messages sizes were in two ranges: large messages with sizes ranging from 50 KB to 80 KB, and small messages with sizes between 1 KB and 8 KB. The traffic could be from various services, such as control, imagery, videos and telemetry; and the sizes used reflect either the size of the whole file or fragmented files at the source node. When generating the messages, each one is assigned a priority 1, 2 or 3 in addition to a TTL value. With these parameters, the number of messages generated in each run varies with an average between 13,000 and 65,000 messages per run. We note that not all generated bundles are routable for various reasons including and not limited to unavailable temporal outgoing edges from the source node after the creation of the bundle. This is dues to the fact that large numbers of messages are randomly generated and cannot be manually checked. Furthermore, there is no oracle method that can specify the exact number of messages for which a route could possibly exist in order to remove the invalid ones.

Table 3.6: Summary of window size for maximal bundle delivery in each network and traffic load.

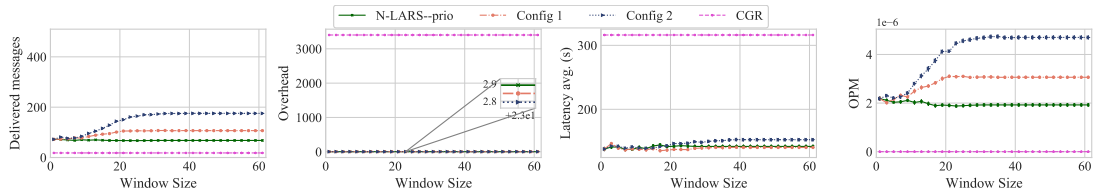
Net size	Arrival rate	MARS Config1	MARS Config2	N-LARS
16	λ=3	37	45	5
	λ=9	37	45	7
	λ=15	33	41	1
26	λ=3	29	31	1
	λ=9	33	45	3
	λ=15	35	43	29
38	λ=3	25	35	29
	λ=9	25	35	31
	λ=15	31	43	13
48	λ=3	17	29	5
	λ=6	39	39	29
	λ=15	39	39	1

3.6.3.3 Numerical Results and Analysis

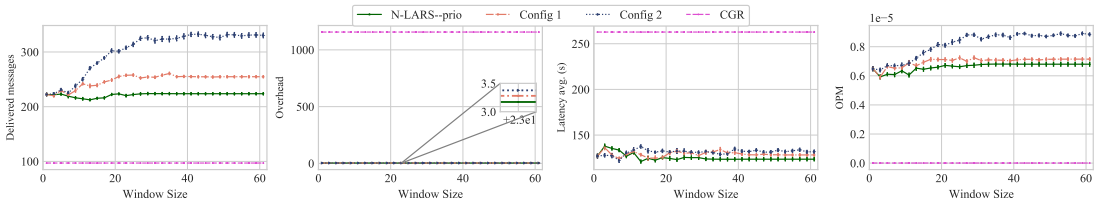
In this section we describe two sets of results. We first present the performance analysis of MARS algorithm and compare it to N-LARS and CGR (when applicable). In these results we use two weight assignment configurations for MARS: $config1 = [size : 0.1,\ ttl : 0.4,\ prio : 0.4,\ tib : 0.1]$ and $config2 = [size : 0.5,\ ttl : 0.1,\ prio : 0.1,\ tib : 0.3]$. In the second part, we present the results of varying MADM combinations and comparing how they affect the network performance.

Varying network and traffic size

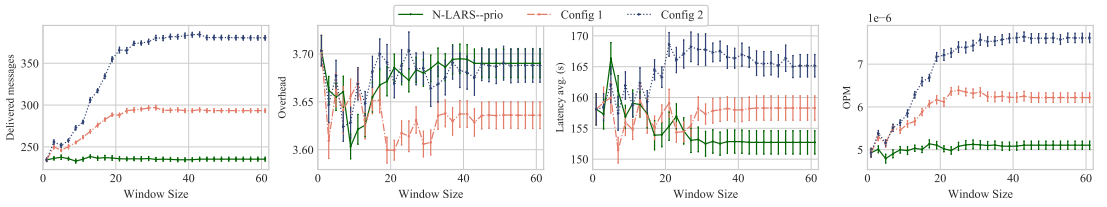
In this section we study the effect of the network size and traffic load on the choice of the window size. In addition, we compare MARS to CGR and our previous scheduling and routing heuristic, N-LARS. We used networks of sizes 16, 26, 38, and 48 nodes and traffic loads described in Section 3.6.3.2. Figure 3.7 shows the results for different network sizes. The graphs in every subfigure present the message delivery count, the overhead, the average latency and the Overall Performance Metric (OPM) from left to right. The results show a consistency across all



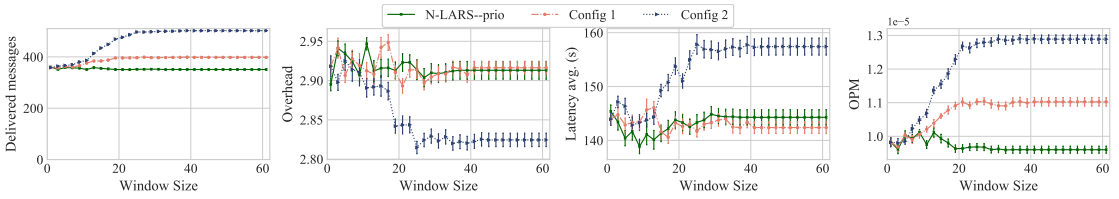
(a) 16-node network.



(b) 26-node network.



(c) 38-node network.



(d) 48-node network.

Figure 3.7: Performance analysis and comparison in different network sizes with $\lambda = 15$.

network sizes in that MARS with config2 outperforms MARS with config1 which in turn outperforms N-LARS with priority. MARS and N-LARS also outperform CGR in the 16-node and 26-node networks. By giving a higher weight to the size and the TiB, MARS with config2 is able to choose the bundles that have smaller size, while still keeping service fairness by prioritizing bundles that stay longer in the buffer. MARS with config1 on the other hand, gives more importance to the

priority and to the TTL, which could prioritize larger and newer bundles to older ones in the buffer. In contrast, CGR showed scalability limitations, which led to the lack of result in networks with more than 26 nodes (or around 5000 temporal edges).

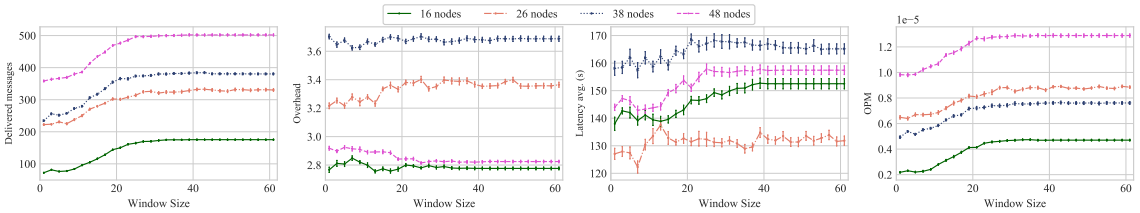


Figure 3.8: All network sizes comparison using config2 and λ=15.

Through diversifying the traffic load and the network size, we drew conclusions on the relationship between these two parameters, the window size and weight matrix configuration. We summarize our findings in Table 3.6. As we expected, the traffic load and network size had an effect on the size of the window that yields the maximum delivery. The table shows that for the same network, as the traffic load is higher, the optimal window size also increases. However, for the same traffic load, as the network size increases, the window size decreases. This is due to the fact that for the same number of bundles, there are significantly more edges to choose for the path; therefore, the number of bundles in the buffer decreases as the network size increases. With that, the needed window size also decreases.

Another conclusion is that by only changing the weight matrix configuration, the number of delivered bundles is significantly improved while still keeping the overhead and the average end-to-end delay as low as N-LARS with priority. This brings a high added value to MARS algorithm since increasing the bundle delivery does not come with much compromise to other metrics, namely overhead and average end-to-end delay. The graphs show, however, that the network per-

formance reaches an maximal point, it enters into a steady state where increasing the window size does not make any difference.

Finally, we compare the behavior of the network using the same weight matrix configurations (i.e. config2) and the same network traffic load (λ=15), but varying network sizes and show the results in Figure 3.8. The graphs in the figure show that after the network performance reaches an maximal point, it enters into a steady state where the size of the window does not make any difference. The graphs in Fig. 3.8 also allowed us to compare the stages at which each network reaches the steady state; a depiction that is complementary to the information in Table 3.6.

Comparing 255 MADM combinations

In order to compare the performance of different weight combinations for the scheduling attributes, we used a network of 26 nodes composed of 4730 temporal edges over 3 days. The traffic was following a Poisson arrival rate of 12 bundles/minute totaling an average of 52050 bundles. We used 255 combinations out of the 282 possible ones (90% coverage) and show the results in the following.

Figure 3.9a shows the number of delivered bundles for different configurations with the break down of the delivery into three levels in the three accompanying graphs. The graphs show that as the weight of the size increases and those of the other three parameters decrease, the delivery improves. Inversely correlated to this, the average overhead improves with the decrease in the weight of the size, as shown in Fig. 3.9b. We note, however, that the difference is not significant. Finally, the graphs in Fig. 3.9c depict the average end-to-end delay which is also not significantly different across configurations.

These results allowed us to find the best configurations for this kind of network

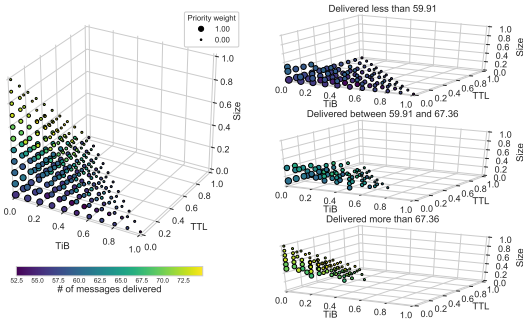
Table 3.7: Weight matrix configurations for the higher delivery ratio.

Size	TTL	prio	TiB
8	1	0	1
4	5	0	1
7	0	1	2
6	1	0	3
6	1	1	2
5	3	0	2
7	0	0	3

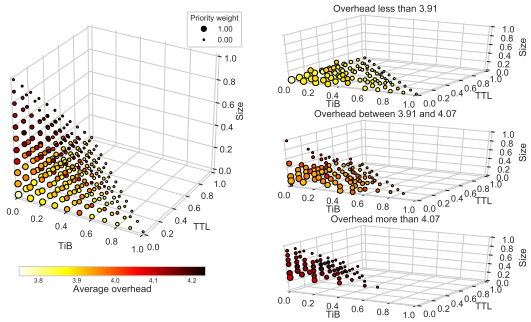
setup and traffic, and we present them in Table 3.7. The choice of the configuration hence depends on what the network application allows us to compromise. If the bundle priority is more important, combinations with higher priority weight are preferred. If fairness in sending bundles is preferred and starvation in the outgoing buffer queue is to be avoided; then configurations with higher TiB weight should be chosen.

3.7 Conclusions and Future Work

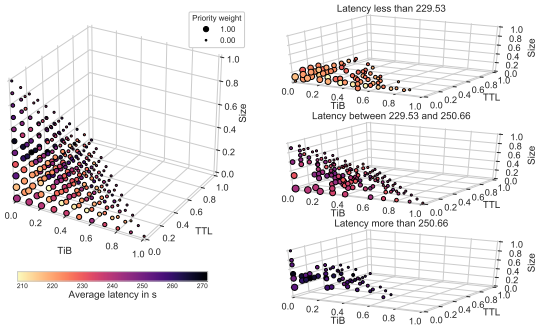
In this chapter, we have introduced the first MILP model for space networks that formally defines the scheduling and routing of bundles using Multi-Attribute Decision Making principles. Since this model does not converge in a reasonable time, we proposed an MADM-based algorithm: Multi-attribute Routing and Scheduling (MARS). MARS algorithm uses a sliding window of size n to schedule the first n bundles in the buffer based on multiple attributes; namely, we used size, priority, TTL and TiB (time in buffer). After finding the optimal schedule for these bundles (in terms of delivery rate), they are routed using our proposed Dijkstra-based routing algorithm. The latter uses the temporal graph model introduced in [6]. Hence, we studied the effect of using PROMETHEE II MADM technique on the performance of the network. Finally, we run multiple simulation experiments in order to evaluate the performance of the proposed MARS.



(a) Number of bundles delivered.



(b) Average overhead.



(c) Latency average.

Figure 3.9: Comparison between 255 different MADM weight matrices.

We compared the performance of our proposed algorithm, MARS, to CGR and N-LARS, proposed in our earlier work [5], with priority using two experimental setups, small-scale Earth-Moon-Mars network and large networks.The results show a consistency across all network sizes in that MARS with config2 outperforms MARS with config1 which in turn outperforms N-LARS with priority. MARS and N-LARS also outperform CGR in the 16-node and 26-node networks, while CGR does not produce any results for the 38- and 48-node networks. In the small-scale network, the delivery ratio was 5.7% better when MARS is used, compared to CGR. The average end-to-end delay was improved by 86% while the overhead was 5 times lower when using MARS algorithm. We summarize the

results for small and large-scale networks below.

In the large-scale networks setup, we used four large-scale networks with sizes ranging from 4476 to 40824 temporal edges each coupled with three traffic loads with a Poisson arrival rates. We compared the performance of MARS to CGR and to N-LARS with Priority. By varying the weights and comparing two different weight matrices for MADM, we proved that the choice of the weights of the attributes do make a significant difference in the performance of large-scale networks. We also showed through our comparison with N-LARS–prio that scheduling based on multiple attributes of the bundles improves the network's message delivery while still keeping the delay and overhead as low as N-LARS–prio.

Finally, by comparing MARS to CGR, we showed that the use of the sliding window enhances network performance compared to the conventional priority-sorted or FIFO based routing. Through these experiments, we also studied the correlation between the message delivery and window size and concluded that the larger the window size the better the delivery is. We note, however, that the network reaches a steady state after a certain window size. In order to further study the effect of the weights on the performance, we run the same sets of network traffic in the same network setup using 255 different weight combinations (90% of all possible ones) and reported the results. The latter gave a better understanding on how the weight assignment affects the performance for that specific network. Not only did it show that the single parameter routing does not yield the optimal results, but it also allowed us to find the right configurations for the best results.

We used a small-scale network composed of 9 nodes all of which are real objects from on-going and future deep space missions. We compared MARS algorithm to CGR on traffic composed of four types of data streams, namely

telecommand, telecommand acknowledgement, telemetry and science. The results showed that while MARS improves the performance of the network in all four metrics used in this work, the role of MADM and the look-ahead scheduling is not significant. Because the bundles are not diverse enough and the routing options are not varied, sorting the bundles does not necessarily show any improvement, but it does not degrade the performance.

The proposed MARS algorithm is not only suitable for deep space scheduled networks, but it is also perfect for many other Earth-bound networks such as Earth-orbiting satellite networks, High Altitude Pseudo-Satellite (HAPS) networks, infrastructure-less disaster-recovery networks and so on. As we have showed that MADM methods, together with scheduling and routing, can significantly improve the performance of DTN-based IPN, a future extension of this work is to apply this combination to other types of DTNs and to hybrid DTNs (i.e. a DTN composed of different types of networks and nodes) in order to enhance their performance. One limitation of the proposed MARS is the extra computation for sorting the messages in the buffer; however, this extra computation results in a great improvement of the network performance. MARS also has some parameters that have to be set ahead of time such as the thresholds of PROMETHEE II and most importantly the choice of the weight matrix, in addition to which attributes are used. These parameters are based on the requirements of the network and the applications that it runs; therefore, it is important that they are input by the user to ensure accuracy and flexibility in setting the network. Finally, in future work, we also plan to incorporate in-network bundle fragmentation and its effects on routing, scheduling and reliable delivery in DTN-based IPNs.

Chapter 4

STAN+PETRA: A Statistical Analysis Aided Routing Algorithm for QoS in Mission-Driven IoT Networks

4.1 Introduction

At the dawn of the 21^{st} century, with the introduction of the term "Internet of Things", the field of networking has been revolutionized. Many applications and enabling technologies have been developed using sensor based edge devices, constituting an ad hoc heterogeneous sensor network, to access network services and transfer real-time data on-the-fly. In this chapter, we study QoS in IoT networks that are deployed to serve a specific mission and which we call MD-IoT networks. There are many examples of such networks, including and not limited to emergency and disaster relief missions such as the one depicted in Figure 4.1 and Agricultural IoT. These networks have multiple common traits, among which we can list the following: (i) they are deployed with a specific mission in mind, (ii) they usually have an underlying ad hoc heterogeneous sensor network for data collection and finally (iii) they require minimal human intervention. Routing in these networks is especially challenging in that it is important to keep the network connected even though its topology dynamically changes over time. Therefore, conventional routing algorithms for opportunistic networks are

limited in overcoming the dynamic spatio-temporal aspects of these networks.

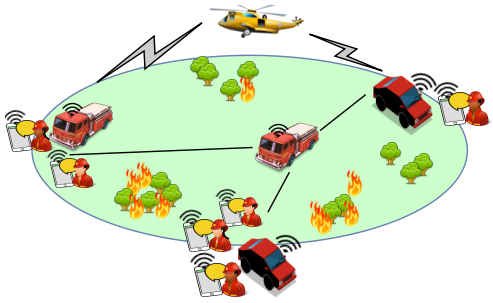


Figure 4.1: Fire department disaster response network.

We *propose the Statistical Analysis (STAN) framework to statistically analyze the MD-IoT network traces* in order to test its predictability and *choose the most appropriate deep learning or traditional prediction model*. We then *design and implement a Dijkstra-based routing algorithm, PETRA, that uses STAN's* prediction in route computation in order to evaluate the effects of STAN on routing. Lastly, through *extensive network simulations using the Opportunistic Network Environment (ONE)* [53] simulator, we show that by using STAN's predictions in routing, we are able to enhance network QoS in terms of reduced overall path delay and increased throughput. We used a dataset composed of real traces collected from a Fire Department to evaluate the performance of our proposed STAN framework and PETRA routing and compared it to the Cognitive Routing Protocol for Opportunistic Networks (CRPO), an algorithm from related work [63].

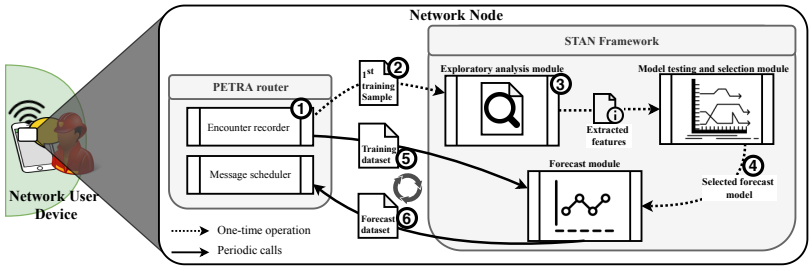


Figure 4.2: System architecture: interaction between STAN and PETRA.

4.1.1 Proposed Approach Overview

The work proposed in this chapter is composed of two main parts: (1) the framework STAN and (2) the routing algorithm PETRA with an integrated prediction model derived using STAN. Time series forecasting has been proved in the literature to improve the quality of network management in various aspects. Though there is a plethora of prediction models to choose from, only a few are suitable for a given application. Furthermore, complex forecast models are not always needed for all datasets; some time series require simpler forecast models than one would choose thinking that a more complex model would necessarily perform better. Finally, deciding about the amount of data history required to increase the accuracy of the prediction can be challenging. Storing more history data then is optimal, would not only have minimal effects on enhancing the performance of the model, but it would also waste precious storage space.

To overcome these challenges and for other benefits mentioned in the subsequent sections, we propose the STAN framework and use it in PETRA routing algorithm as depicted in Figure 4.2 and detailed in the following steps numbered as in the figure:

1. Encounter Collector of a node collects its encounters with other nodes and any traces these nodes share with it.

2. The node's recorded traces are used as tuning and training for STAN framework to perform better.

3. The exploratory analysis module extracts characteristics of the dataset to define forecast models' hyperparameters.

4. STAN's model testing and selection module chooses the right forecast model for the collected traces.

5. The forecast dataset is returned to the routing algorithm for route computation.

6. New collected traces are used for further forecasts with no need for global knowledge of the network.

The STAN framework is detailed in Section 4.3 and PETRA routing algorithm is described in Section 4.4.

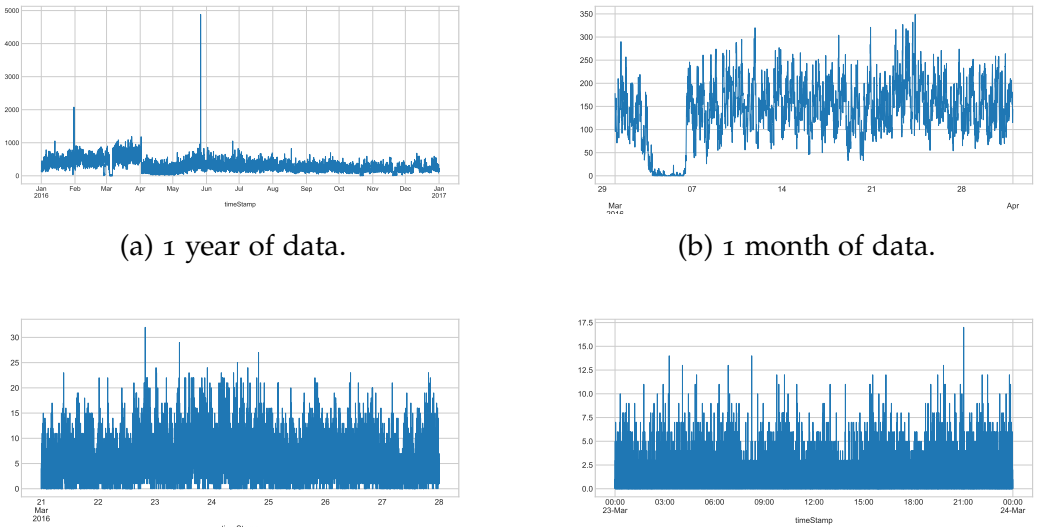
4.2 Related Work

The concept of IoT is very broadly defined and can support a variety of applications with many different constraints and requirements. Therefore, QoS in IoT is contingent on the application. A common trait among all these applications, though, is that forecast/prediction methods are very commonly used to predict QoS attributes in order to improve it [64]. White et al. [65] proposed a QoS prediction method to enhance the performance of IoT service composition. They model the QoS of the network in terms of throughput and response time, and predict the QoS for possible services based on service usages and do not provide flexibility in prediction methods. Time series forecasting has also been exploited by multiple techniques implemented to improve the quality of service in VoIP, VToIP/IPTV and WWW applications as detailed in [66]. In [67], Mu et al. look at QoS in IoT in terms of radio and transmission power selection. Their proposed solution incorporates multiple wireless technologies to enhance the QoS of the network under the constantly changing environmental conditions.

Similar to the aforementioned example applications of IoT, QoS in the spatio-temporaly dynamic MD-IoTs is determined by the availability of the spatio-temporal edges between two nodes, the overall path delay that absence of edges causes and the network QoS. Zdarsky et al. [68] addressed the problem of intermittent connections caused by the movement of nodes through the concept of movement contract. Each node in the network publishes its movement to the network so that the other node anticipates the link changes. This solution is not practical for the MD-IoT as the movement of the nodes is the norm and not the exception, which would cause too much overhead for the network. In [69], Duan et al. proposed a multi-layer QoS architecture for IoT. As such, they gather requirements from the application, network and perception layers and use them in the definition of their architecture. This generalized architecture could be applied to different common applications; however, because it is very broad, its QoS attributes do not cover the spatio-temporaly dynamic aspect of the MD-IoTs studied in this chapter. Last but not least, an IoT QoS model was proposed in [70]. The authors studied the effect of the limited storage capacity and processing power on the QoS in IoTs and proposed a model to enhance it. The model is proved to outperform the Best Effort service model, but was not applied to IoTs with spatio-temporal constraints. Therefore, the underlying assumptions for building this model are not applicable to MD-IoTs. In this work, we study the MD-IoT network traces and propose a STAN Framework and PETRA routing algorithm that improve the QoS of these networks. The general architecture and advantages of the framework is described in the following sections.

4.3 The Statistical Analysis Framework: STAN

The framework has two main modules. The first module performs an exploratory analysis of the time series. It provides information on the attributes of the dataset such as stationarity, seasonality and so on. This information is important for the second module, which carries out model selection. We use the oviedo/asturies-er Dataset, described in the following subsection, to demonstrate the operations of the framework.



(a) 1 year of data.

(b) 1 month of data.

(c) 1 week of data.

(d) 1 day of data.

Figure 4.3: Temporal portioning of oviedo/asturies-er traces.

4.3.1 The Test Case Network: The oviedo/asturies-er Dataset

The oviedo/asturies-er (v. 2016-08-08) dataset [71] is composed of real traces from the Fire Department of Asturias, Spain. The network was composed of a helicopter, and multiple trucks, cars and personal radios, totaling in 229 devices. The traces were collected in both emergency interventions and in the daily routine. There are three traces, and for our test case, we use the traces assuming connec-

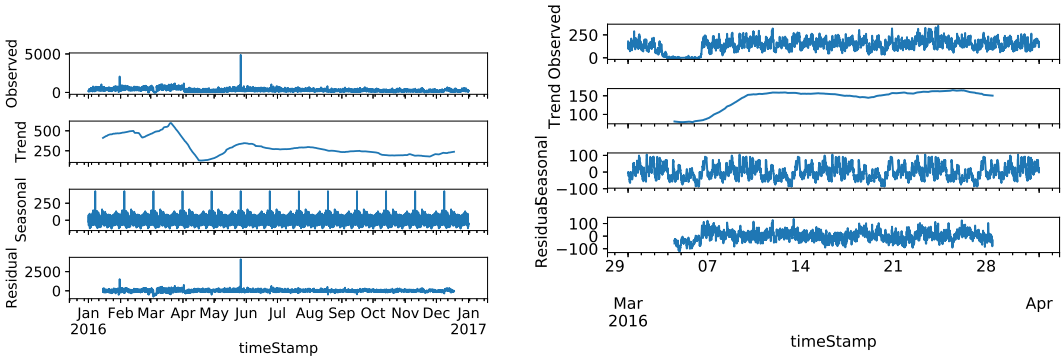
tivity within 50 meters of separation between every two nodes. The dataset is composed of one year worth of traces, and without loss of generality, we assume that the recording started at midnight of the 1st of January in 2016.

4.3.2 STAN's Exploratory Analysis

Exploring the data before building a model is very important in that it allows to better understand the problem at hand, assess the usefulness of the data, and accurately define the objective of the prediction and hence selecting the forecasting model with the least complexity for the given dataset. Analyzing time series uncovers the effect of time on the data and permits the framing of the problem with the consideration of time. Finally, data can be better fitted and pre-processed for the chosen forecasting model. STAN's exploratory module starts by preparing the data. That is, the time series is partitioned by pairs and by temporal attributes. In the example oviedo/asturies-er dataset, the time series is temporally divided into 4 samples: 1-year long, 1-month long, 1-week long and a day long parts as depicted in Figure 4.3. The graphs show the frequency of encounters across all the pairs of nodes in the network (6576 pairs in total). We also extract traces for a sample pair for one year, one month, one week and one day. In this example there are two attributes that we look at in the analysis of pairs: frequency of encounters and the duration of encounters. We focus on three major traits of data, each of which is explained next.

4.3.2.1 The Five Number Summary and Box-and-Whisker plots

In order to identify the distribution of data and properly compare it to other samples, we integrate the widely used Tukey's 5-number summary [72]. The summary is composed of the minimum value, the first quartile representing the first



(a) 1 year of data with one month frequency.

(b) 1 month of data with one week frequency.

Figure 4.5: Additive decomposition of oviedo/asturies-er traces and samples.

number greater than 25% of the datapoints, the 2^{nd} quartile which is the median, the 3^{rd} quartile (similar to the 2nd quartile but represents 75%), and the maximum value. These values show how data is distributed around the median and to which side it could be skewed. It also allows to identify outliers and to compare the distribution of the samples. Figure 4.4 depicts the box-and-whiskers plot of the frequency of encounters in three different months. It shows that the distribution of February and March are very close to each other, while the distribution of April is significantly different and has many outliers.

4.3.2.2 Decomposition for Stationarity and Seasonality

Stationarity refers to whether the time series has a trend or not. Many forecasting methods do not perform well on data that has a trend because they assume traces are stationary and hence either result in overfitting or underfitting. Seasonality is also an important feature in that it can be leveraged when forecasting time series that are effected by time. For example, the communication pattern of one day could be repeated throughout the week. Both trend and seasonality can be removed allowing forecasting models to better perform. There are, how-

ever, prediction models that are not affected by trend and seasonality. Therefore, it is important to identify whether the time series has these attributes. STAN framework uses decomposition as shown in Figure 4.5. The graphs show the decomposition of one year and one month samples of data. The figures show trend and seasonality in the data samples. The year has a monthly seasonality, the month has a weekly seasonality and so on.

4.3.2.3 Distribution in the Quantile-Quantile (QQ) and Probability Plots

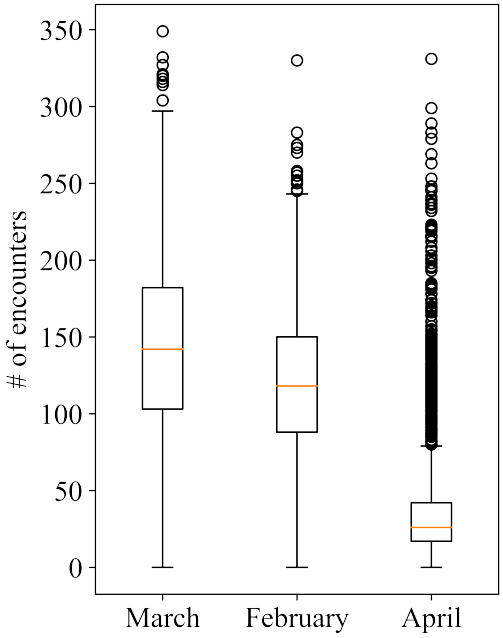
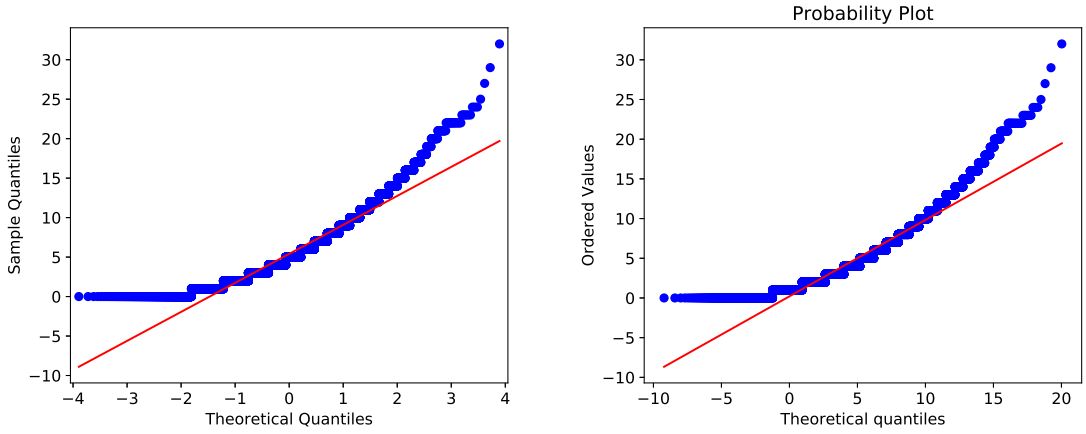


Figure 4.4: The box-and-whisker plot comparing three months.

In addition to the five number summary, our proposed STAN framework extracts times series quantiles in order to compare the time series' distribution to other well known distributions such as Normal, Poisson and Gaussian. For that we use the quantile-quantile (QQ) plot and the probability plot. The QQ plot compares the quantiles of the time series to equal quantiles of a distribution in order to determine whether they match. Figure 4.6a and 4.6b, show the QQ-plot and the probability plot, respectively, of the week of data depicted in Figure 4.3c. In these figures, the week traces are compared to the Normal distribution and it clearly shows that the data does not follow a Normal distribution since the blue points do not fall on the line.



(a) QQ Plot.

(b) Probability Plot.

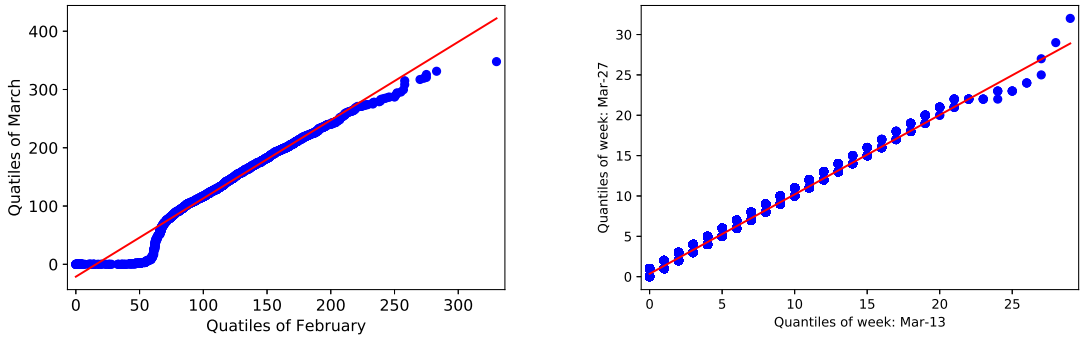
Figure 4.6: QQ-plot and the probability plot of one week of traces.

4.3.2.4 Comparative Distribution: The 2-Sample QQ Plot

While the QQ-plot allows us to compare the distribution of the time series to well known distributions, sometimes it is more useful to compare the distribution of two data samples in order to better understand the data at hands. The 2-sample QQ plot allows us to do that. We compare two partitions of the same size in order to verify the consistence of the distribution across different samples. An example is shown in Figure 4.7. Figure 4.7b shows that two weeks of traces have very similar distributions; in contrast, Figure 4.7a shows that two months do not necessarily have similar distributions. This means that the forecast unit has to be well chosen for this data set. That is, if the forecast is done for one month ahead, the performance might not be as good as if the forecast is for one week at a time.

4.3.3 STAN's Model Testing and Selection

The framework includes five forecast models composed of two classical prediction methods (Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA)), and three deep learning methods, namely Convolu-



(a) Comparing two months. (b) Comparing two weeks.

Figure 4.7: 2-Sample QQ plot of one month and one week of traces.

tional Neural Network (CNN), convolutional Long Short-Term Memory (convLSTM) [73] and Multi-Layer Perceptron (MLP).

4.3.3.1 Classical Forecast Methods

Including classical forecast models comes with the advantage of simplicity of computation and speed in obtaining results. Therefore, ARIMA and SARIMA are included in the STAN framework. The ARIMA method is based on a linear function of differenced observations and residual errors of the history of traces. Hence, this method is only suitable for time series that do not have seasonality and deals with trend by differencing the time series in order to make it stationary. Besides the computational simplicity, ARIMA

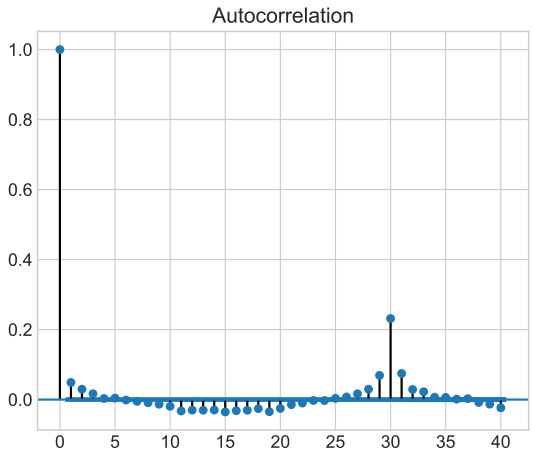


Figure 4.8: Autocorrelation of a day of data.

is also advantageous in terms of the number of hyperparameters used for the predictions. In fact, it takes three parameters to model the trend: (1) the autoregressive order 'p' that reflects the autocorelation between different lags of the time series, (2) the difference order 'd' representing the trend of the time series and (3) the trend moving average 'q'. These parameters can be inferred from STAN'S exploratory analysis module as also shown in Figure 4.5 and Figure 4.8.

Seasonality can be removed using the difference transform, which would make the time series suitable for ARIMA; however, STAN includes SARIMA to avoid this extra step. As its name suggests, Seasonal ARIMA uses the same linear function as ARIMA, but adds to it the differenced seasonal observations and seasonal errors of prior traces. SARIMA takes four extra parameters to model the seasonality: (1) 'P' for the seasonal autoregressive order, (2) the seasonal moving average order 'D', (3) 'Q' for the seasonal moving average order and (4) 'm' representing the number of time steps in each season (e.g. one week per month). These parameters can also be extracted from STAN'S exploratory analysis module and are depicted in Figure 4.5.

4.3.3.2 Deep Learning Methods

Applying Neural Networks (NN) for deep learning provides high flexibility and has been shown to perform well on various types of problems. A variety of algorithms have been proposed with an overarching architecture consisting of: an input layer, one or more hidden layers and an output layer. We chose to use three of the implementations: CNN, convLSTM, and MLP. CNN is one of the commonly used neural network algorithms in various applications and especially in image processing. Unlike ARIMA and SARIMA, CNN's performance does not deteriorate by the presence of seasonality and trend. Furthermore, CNN

algorithms are effective in dealing with noise and can reduce long time series into small feature maps that they then use to learn important features for the forecast. Likewise, MLP maps inputs to outputs through its hidden layers in order to predict future values. It is also well suited for time series. Last but not least, convLSTM was specifically proposed to model the spatio-temporal relationships in a time series. As such, it has multiple benefits for our application including stability. Details of convLSTM can be found in [73].

The implementations of CNN, MPL and convLSTM used in STAN allow the tuning of at most four hyperparameters. These are (1) the number of parallel *filters* used as feature detectors, (2) the *kernel size* which determines the number of time steps used in each input sequence, (3) the *epochs* representing the number of time the whole training dataset would be exposed to the model and (4) the *batch* size as the number of examples from the training dataset presented in a single batch (i.e. the training dataset is divided into multiple batches). Optimal values for these hyperparameters depend on the features of the times series and the size of the multi-step forecast desired, among others. And setting larger or smaller hyperparameters can affect both the prediction accuracy and the execution time of the algorithm. And some of them cannot be inferred directly from the data. Therefore, grid searching the hyperparameters is also included in the proposed framework and described in the following section.

4.3.3.3 Grid Searching Hyperparameters

The exploratory analysis module of our proposed framework STAN helps determine most of the hyperparameters needed for the aforementioned forecast models. Nevertheless, some time series may have multiple optimal values for each hyperparameter. Therefore, STAN's model testing and selection module includes

a huperparameter grid searching functionality, which takes multiple values for each hyperparameter and finds the most optimal combination in terms of prediction accuracy. In order to reduce the complexity of the grid search, the exploratory analysis module can be used to reduce the options for potential hyperparameter values.

4.3.4 STAN's Forecast Module

This module generates the forecast based on the selected model and the training traces provided by PETRA's encounter recorder. It is the most frequently invoked module by PETRA. This module is implemented using the ensemble Gradient Boosting for classification (GBC) and the model selected by MTS module in order to incorporate binary classification and enhance the forecast. That is, the model selected by the MTS forecasts traces for a duration Δt as list of pairs $(time, duration)$. This forecast is then used by the GBC classifier to provide the probability that the listed encounters would actually occur. GBC is based on regression trees and are used to improve the predictions' accuracy.

4.4 The Predictive Routing Algorithm (PETRA)

In the next section, we describe the proposed routing algorithm, PETRA. In addition to the predictive model integration, PETRA is also made network characteristics-aware by treating different types of contacts according to their characteristics. The term contact is defined as a time window in which two nodes can exchange data and has been used in satellite communication networks' routing [6, 35]. Broadly speaking, communication can happen through already scheduled contacts that are usually pre-programmed in the communicating devices. It can also happen

through continuously available contacts such as the Internet. Finally, communication can take place using discovered contacts in the network and these are the most common in MD-IoT. PETRA is designed to support all these kinds of contacts in order to enhance its flexibility and performance.

4.4.1 The PETRA Routing Algorithm

Our routing algorithm is designed to route messages based on the three cases introduced above; therefore, it has three main attributes. We first distinguish between a scheduled path and a hybrid path. The former is a path that is composed of solely edges extracted from the provided schedule (scheduled contacts) and the second can be composed of a mixture of scheduled contacts and the ones discovered and predicted using STAN's predictive mode. We then add a source route to each message. This message property can either be used or left null. Finally, we define the path expiration time as the time at which a path has to be re-computed to account for network changes. Furthermore, the path is assigned a probability since the predicted edge would occur with a certain probability. Communication windows extracted from the schedule are assigned a probability of one. The routing algorithm is detailed in Algorithm 4 and described in the following along with its properties.

Each node that creates a message computes its source route. In order to add the source route to the message, the computed path has to be scheduled. If the path is not scheduled (i.e. its aggregate probability is less than 1), there is no need to add it to the message for efficiency since the network is likely to change fast enough to invalidate the source route. PETRA starts by checking for a source route ($srcRoute(m)$). If it exists, it is checked for expiration. That is, if a certain amount of time passes after the route was initially computed, it

Algorithm 4 PETRA Routing Algorithm

Input: Temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$ in its edge stream representation, message m
Output: Final path and routing decision
if $srcRoute(m) \neq \emptyset$ and $srcRoute(m).isNotExpired$ **then**
    Send m to next connection in srcRoute(m)
else
    $newRoute \leftarrow buildPath()$
    if $newRoute.isScheduled$ **then**
        $srcRoute(m) \leftarrow newRoute$
        Send m to next connection in newRoute
    else if $proba(newRoute) > threshold$ **then**
        Send m to next connection in newRoute
    else
        $m.keepInQueue \leftarrow true$

expires allowing the node to reroute it using more timely information about the state of the network. Hence, if the message source route has expired, it has to be computed again and replaced. Otherwise, the source route is used and the message is sent to the next temporal edge as specified in the source route.

In case the message does not have a source route or the latter has expired, the current node (either a relay node or the source itself) computes a new path, *newPath*, using the path computation algorithm described in Algorithm 1 in the next section. If the new path is scheduled, it is set as the source route of the message and the latter is sent to the next contact as specified in the route. Otherwise, it checks its aggregate probability computed as $\prod_{\forall e \in path} \beta(e)$, the multiplication of the probability, β, of all the edges in the path. If this value is greater than a threshold, the route is used and the message is sent to the next node in the path; otherwise, the message is kept in the queue until a better path is found.

Finally, we defined two modes of operation for PETRA. In the standard operation mode, which we refer to as PETRA for simplicity, a message is sent to the next

node according to the predicted edges in the path. That is, if the next predicted edge starts at time t_i, then the message is sent through the same edge if it occurs at time $t_i \pm \Delta t$. In the second mode, *which we call aggressive mode (PETRA-am)*, the start time of the edge is disregarded, and the message is forwarded as soon as a contact with the next node in the path takes place.

Algorithm 5 *buildPath* method

Input: Temporal graph $G_t = (\mathcal{V}, \mathcal{E}_t)$ in its edge stream representation, source s, transmission start time t_s, TTL, data total size D_s, v_{dst} the destination node
Output: The final set of paths *PathSet*
$PathSet \leftarrow \emptyset$
$Visited \leftarrow \{s\}$
while $Visited \neq \emptyset$ **do**
  if s is the destination **then**
    if path is scheduled **then**
      $PathSet \leftarrow path$
      return
  else
    Init: $\mathcal{N}(s)$ ▷ Initialize neighbor-set of s to all its known neighbors
    $\mathcal{N}(s) \leftarrow f : STAN() \cup \mathcal{N}(s)$ ▷ Find all the temporal edges $e_{(s,v)}$
    if neighbor-set $\neq \emptyset$ **then**
      for all $v \in \mathcal{N}(s)$ **do**
        Remove all edges with $p(e) < threshold$
        $route(E_v)$
    $Visited \leftarrow Visited \setminus s$
    $s \leftarrow next(Visited)$
return *PathSet*

4.4.2 Route Computation

The route computation is performed by the method $buildPath()$ detailed in Algorithm 5. It starts by initializing the variables described in Table 4.1 to the values shown in the same table. Then for every node 's' in the set *Visited* (the set of nodes processed by the algorithm), the algorithm checks if 's' is the final destination in

order to save the path and return it. Then it checks if the set of neighbors of 's' is not empty in order to continue the path computation. If not, for every node v in the neighbor set of 's', the temporal edges are computed using STAN and added to the neighbor-set of the node 's'.

Choosing the next best node, n, is based on two criteria: the category of the contact and then its probability β_{agg}. β_{agg} refers to the probability of reaching n and is obtained by multiplying the probabilities of all the temporal edges that lead to n. The scheduled edges are preferred to the predicted ones. Therefore, if a node in *visited* set is scheduled it is chosen; otherwise, the node with the highest probability is chosen as the next best. β_{agg} and *visited* sets are both determined using the relaxation function $route()$ in line 16 of Algorithm 5.

Table 4.1: Variables used in Algorithm 5.

Variable	Description	Init. value
PathSet	The set of path to be returned at the end	$\emptyset$
Visited	The set of nodes that have been already assigned a cost value	$\{s\}$

Algorithm 6 route Algorithm

```
1: Input: The set of neighbors E_v
2: for all e ∈ E_v do
3:    Price_e = computeDP(e, v);
4:    if v ∉ Visited then
5:       if e is (Discovered or Continuous) then
6:          update(v);
7:       else if overlapEdge(e, (t_s, t_s + TTL)) then
8:          update(v);
9:       Visited ← Visited ∪ {v};
```

The method $route(v)$, detailed in Algorithm 6, updates the node v as follows and uses the methods described in Table 4.2 for clarity. For all the edges in the set

Table 4.2: Methods used in Algorithm 6.

Method	Input	Description : output
overlapEdge	Edge e $(t_s, t_s + TTL)$	Checks if the edge overlaps with the time window $(t_s, t_s + TTL)$: True or False
price	v	Retrieves the cost associated with v : (v_{delay}, v_p)
computeDP	v	Computes the cost in terms of delay and probability to reach v : (v_{delay}, v_p)

of temporal edges E_v, it computes the cost to get to v through edge e in terms of delay and probability. Then, if v already has a cost value, the latter is compared to the newly computed price and the best is chosen; otherwise, v is updated with the new value. Finally, the method $update(v)$ is used to set (1) the new parent node for v to be s, (2) the delay as the time it takes the message to reach v using this edge and (3) the probability that this path will occur.

4.4.3 Edge Prediction Integration

The edge prediction is integrated from the STAN framework. STAN is implemented in Python and runs as a stand-alone application that is invoke by the router in each communicating device. As shown in Figure 4.2 and described in Section 4.1.1, STAN reads in a history of encounters or a sample of traces from the network and runs the exploratory analysis module. Data is input as a list of edges represented as a pair of nodes, edge start time and edge duration. Results from the exploratory analysis module allow STAN to determine ranges of values for the hyperparameter grid search run by the model testing and selection module. The latter selects the appropriate forecast method among the 5 implemented (e.g. it discards ARIMA if the time series is seasonal) and builds their models.

The models are scored using The Root Mean Squared Error (RMSE) and the

one with the lowest value is selected. The grid search step can be run at the beginning and/or when the size of the training set changes. That is, if the node always uses 6 hours of traces to predict the next 30 minutes of encounters, the grid search has to be run only once, using a training set of 6 hours. However, if for example, the training set increases by two hours every time the node stores more history of encounters, the grid search has to be run every two hours in order to used the two extra hours for hyperparameter tuning. Finally, the chosen model is passed on to the forecast module, which performs a multi-step forecast. The latter is further adjusted using the GBC. The forecast traces, a list of triplets: time, duration and probability, are relayed to the router of the node. The latter uses the predicted schedule to route messages across the network.

4.5 STAN+PETRA Experimental Setup and Results

In this section we describe the experimental process and present the results obtained from various simulations.

4.5.1 Implementation of STAN and PETRA

In order to evaluate our STAN+PETRA framework, we implemented the algorithm described above in the Opportunistic Network Environment (ONE) simulator [53]. For comparison, we use PETRA's two modes of operation in addition to CRPO [63] as implemented by the authors of the paper and using the default values for the routing algorithm variables. In the experiments, we use 10 nodes running Linux OS in a cluster hosted at a computing center. Each node is an Intel Xeon E5-2670 with 2.60GHz 2 CPU/16 cores and 32 Gb of RAM. We run ONE v1.6.0, the latest version. In all experiments detailed below, for the same network

topology, we route the same traffic setting three times by varying the routing algorithm. Finally, we compare the performance of these algorithms based on the average delay, network throughput, and a combination of latency and delivery ratio (deliv. ratio) which we define as the overall performance metric (OPM) computed as:

$$OPM = Avg.\ Delay/(Deliv.\ ratio \times avg\ hop\ count) \tag{4.1}$$

4.5.2 Experimental Setup: oviedo/asturies-er Dataset

In this experiment, we use the oviedo/asturies-er trace [71] composed of real traces from the Fire Department of Asturias, Spain. The network was composed of a helicopter, and multiple trucks, cars and personal radios, constituting 229 nodes with a total of 443,978 contacts and runs for a duration of 2,309,499 seconds (used in the experiments as ≈15 days of encounters and new messages + 10 days of encounters only). Figure 4.3 shows the frequency of encounters across all the pairs of nodes in the network (6576 pairs in total) for one year. The simulated traffic follows a Poisson distribution, and in order to simulate different traffic loads, we use different arrival rates λ in multiple simulations. Therefore, we run 15 simulations per routing algorithm per arrival rate; that is, we generate 15 different traffic files for every value of λ and run all algorithms using each one of them. The results of the experiment are described next.

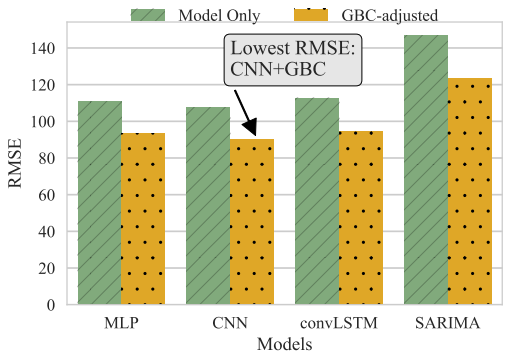
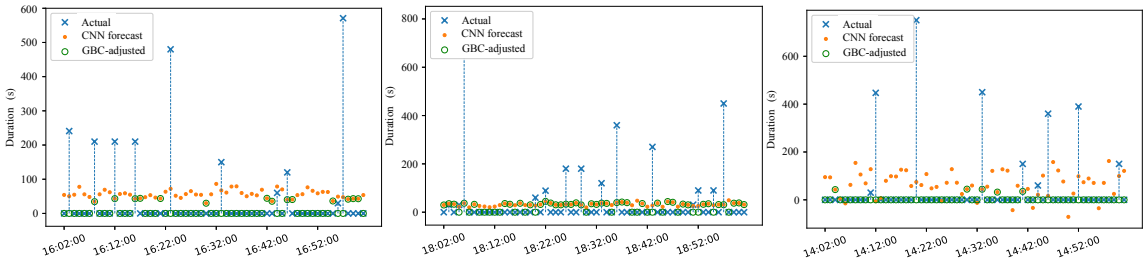


Figure 4.9: RMSE of the four models.

4.5.3 Numerical results and analysis

In the experiment, we assume there is a sample of network traces that is used for model selection and tuning. We use an extract of the dataset for that purpose. Thenceforth, throughout the experiment, each node invokes the STAN framework every simulated hour. The node provides STAN's forecast module with the history of traces and receives the generated forecast as the contact list for the next hour. In the following, we demonstrate the numerical results for both STAN and PETRA.

4.5.3.1 STAN's Forecast Accuracy



(a) Good forecast. (b) Overfit forecast. (c) Underfit forecast.

Figure 4.10: CNN and GBC-adjusted one hour forecast for one pair of nodes.

We start by running STAN on one day of data which is not included in the experiment in order to extract attributes of the traces. Using these attributes, we run STAN's grid search for CNN, MLP, SARIMA and LSTM. ARIMA is not included since it is not applicable for seasonal time series. Figure 4.9 compares the Root Mean Squared Error (RMSE) of the models and shows that CNN is the best model for times series used. Figure 4.10, illustrates the performance of CNN and the ensemble GBC+CNN. The blue crosses represent the actual traces, while the orange dots reflect CNN's forecast and the green circles are results from the ensemble GBC+CNN (CNN's forecast adjusted using GBC). For every hour of

predicted contacts, CNN uses 12 hours of training data for every pair. For some pairs, 12 hours of training data may contain a small number of contacts (e.g. less than 5 per hour); therefore, the forecast is either overfit or underfit as depicted in Figures 4.10b and 4.10c, respectively. Approximately, 53.5% of the pairs of nodes in the network had fewer than 10 contacts throughout the whole simulation time. However, for the remaining 46.5% of the pairs making up 98.1% of the traces, CNN's performance was predominantly well fitted with a performance similar to the sample shown in Figure 4.10a. We should note that because the main purpose is to predict the likelihood of the occurrence, underfitting the duration forecast is acceptable. Furthermore, when using the forecasts in PETRA, we consider the predicted contact to have a duration that is equal to or greater than the predicted duration.

4.5.3.2 PETRA's Numerical Performance

In this section, we summarize the results obtain through running the standard PETRA, the aggressive mode PETRA-am and CRPO. The results focus on the two main metrics for QoS in the MD-IoT networks studied in this chapter: (1) throughput and (2) end to end path delay. Figure 4.11 presents the daily throughput of all three routing algorithms. PETRA's aggressive mode resulted in the highest

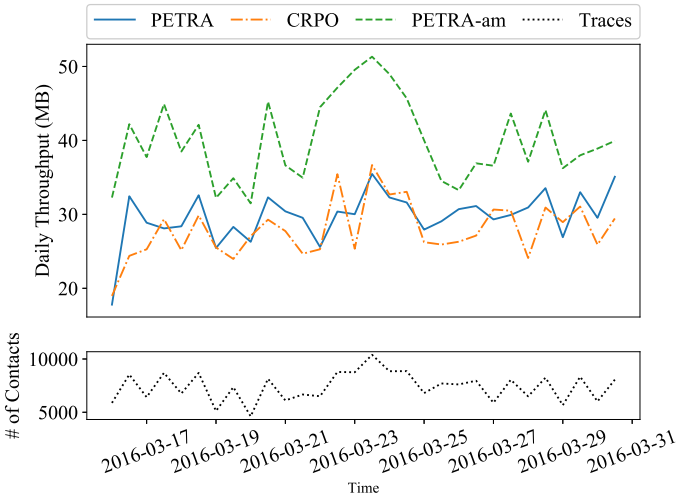


Figure 4.11: Daily Throughput.

throughput over the whole period of the experiment by an average of 29.4%. PETRA enhanced the performance by 4.6% compared to CRPO. The lower graph in the figure displays the number of contacts per day and explains the variation in the throughput in different days. Figure 4.12 depicts the distribution of the end to end path delay. The box and whisker plots show that CRPO and standard PETRA have similar distributions with equal means while PETRA-am shows delay values that are skewed towards larger delays. They also all have a large number of outliers, which is due to the changing number of contacts as discussed before.

Using the aggressive mode allows the network to deliver a larger number of messages while compromising the end to end delay. The average delivery ratio is depicted in Figure 4.13a, which summarizes the graph shown in Figure 4.11. By sending more messages, PETRA-am causes extra delay for individual messages but enhances the overall network performance by scoring the lowest delay units per message (i.e. Equation 4.1) of 0.099s/msg. While PETRA results in better throughput and still maintains a reduced end to end delay, its delay units per message measure was higher than PETRA-am by 0.056s/msg. This means that with PETRA-am each message caused 9.92% of the overall network delay, PETRA resulted in 15.6%. These results are presented in Figure 4.13b.

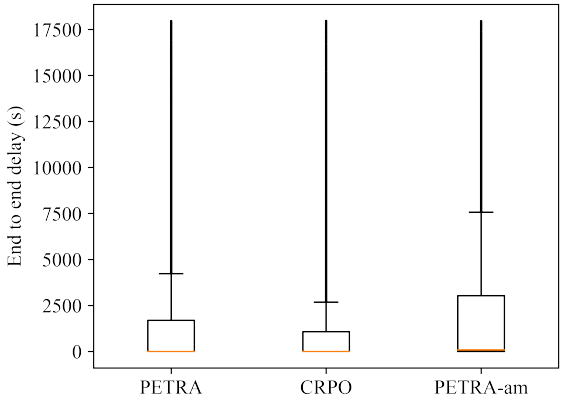
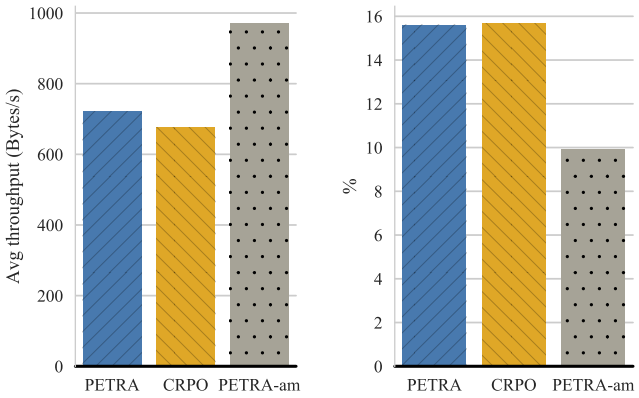


Figure 4.12: End to end delay distribution.

4.6 Conclusions



(a) Throughput summary. (b) The OPM metric.

Figure 4.13: Performance metrics summary.

MD-IoT networks are networks deployed with a mission in mind, such as emergency and disaster relief missions, mega-farms and agricultural applications and deep space colonies and exploration. These networks have an underlying heterogeneous sensor base composed of a variety of devices such as UAVs, personal devices, and vehicles. The work proposed in this chapter aims at improving QoS in MD-IoTs through predicting the dynamic spatio-temporal link availability, reducing overall path delay and increasing the throughput. We proposed a statistical analysis framework, STAN, to represent traces of the network as time series, statistically explore the time series, test deep learning and classical forecasting models, and select the right one for the MD-IoT. Furthermore, we proposed PETRA, the PrEdicTive Routing Algorithm, that integrates the forecasting model selected by STAN. Thanks to data exploration module and the model selection, using STAN allows the routing algorithm to use the prediction model with the right level of complexity and the right amount of history data for training. To the best of the authors' knowledge, this work is the first that improves MD-IoT's QoS by integrating a statistical analysis module with a multiple choices of prediction models and a routing algorithm.

The STAN framework was tested on the real world traces, oviedo/asturies-er dataset, from the fire department of Asturias, Spain. Results showed that the providing various forecast models improves the prediction accuracy. Furthermore, they proved the importance of using the GBC ensemble, which uses CNN's forecast. Nearly 98.1% of the dataset constituted traces that could be used by STAN's forecast module. Through extensive simulations our PETRA routing algorithm was proved to improve the QoS of the network compared to CRPO. As shown in Sec. 4.5, our routing algorithm was able to enhance the network throughput by up to 29.4% while maintaining low end to end delay. We also used the overall performance metric defined as: $Average\ Delay/(Delivery\ ratio \times avg\ hop\ count)$, which reflects the percentage the delay caused by each message and takes into account path diversity. The results showed that the modes of PETRA improved the network performance by up to 36.7%.

Chapter 5

A Low-Latency Routing and Edge Computing Framework for Heterogeneous Wireless Mission-Driven IoT

5.1 Introduction

Mission Driven-Internet of Things (MD-IoT) environments, such as ones deployed in agricultural fields or in smart cities, are prominent applications for the large-scale heterogeneous networks that involve collecting tremendous amounts of data, processing it in a near-real time manner, and sending actionable information to actuators that will act based on it. This process can only be implemented through the use of large heterogeneous networks, which incorporate various aspects of heterogeneity introduced in Chapter 1. These networks need to also integrate data processing mechanisms suitable for different types and amounts of data. For example, in agricultural IoT networks (Ag-IoT

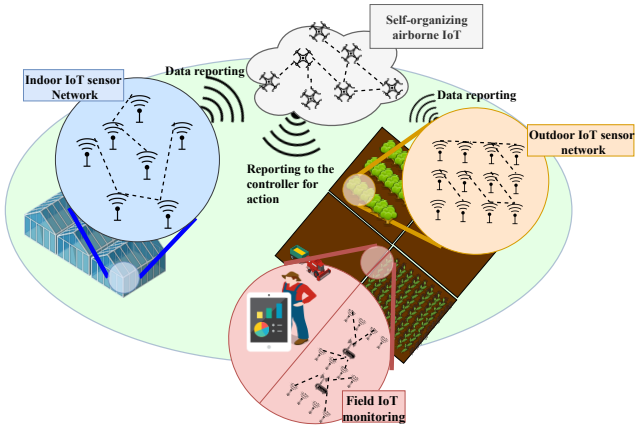


Figure 5.1: Illustration of a mega-farm monitoring Ag-IoT network.

in short), building a robust network architecture has the potential to help increase the yield and better manage resources like water, nutrients, soil, and fertilizers. The questions raised in this context are about how data can be collected and routed efficiently and effectively to processing nodes, what kinds of processing are required for different use cases, how to communicate data back to the actuators, and how to deal with the high dynamicity of the network topology.

Consider a network composed of sensors, relay nodes and processing nodes at an edge computing platform as depicted in Figure 5.2. Each sensor node collects data to be processed at one of the processing nodes. The sensors can send the data to the processing nodes via the relay nodes.

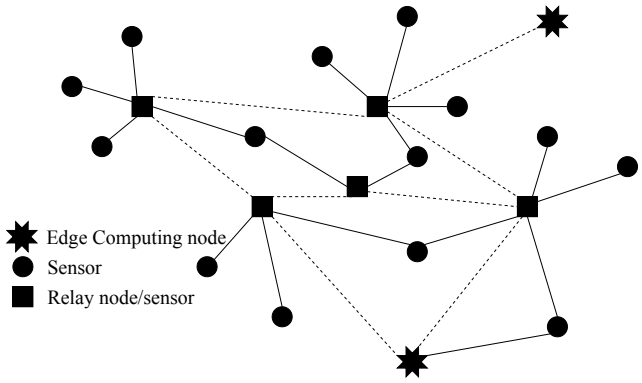


Figure 5.2: Network Topology.

Each sensing and relay nodes has limited amount of energy that it can use for all its tasks (i.e. sensing, communication ...). Each edge computing node has a limited processing power that it can dedicate to a certain task at a time which means that tasks arrive to a queue or are dropped if the system is full. In this work, we build a network model to optimize the routing and processing of data ensuring data freshness, minimal delays and maximum data delivery.

5.2 Background and Related Work

5.2.1 Routing and Computing for IoT Networks

Routing and conputing are two important topics that have been heavily studied in wireless and IoT networks. In this section, we list some of the related and relevant works.

5.2.1.1 Routing in Resource Constrained IoT Networks

The concept of a sink was initially proposed to collect data from a number of connected sensors when a CPS was composed of relatively smaller numbers of sensors and data sources. The static sink collects data and either relays it to the end user or performs basic processing before sending it to a final destination. This paradigm had many limitations as discussed in [86,87] and hence mobile sink architectures were proposed [88–92]. Meanwhile and as the technologies have been evolving and as the number of data sources (sensors, end-user devices and so on) has been increasing, solutions such as Cloud computing, Edge computing architectures, hybrid Cloud and Edge architectures and more have focused on satisfying the need for efficient computational power and scalable storage changing the focus of routing to the timely delivery of data to the compute platform while preserving intermediary nodes' energy. works such as [115–117]

5.2.1.2 Computing in Resource Constrained Networks

Cloud computing has been widely studied and many architectures have been proposed [93,94]. Following that, different research applications have been demonstrated. Resilience in cloud infrastructures has been studied as Colman-Meixner et al. have shown in their survey [95]. Loutas et al. surveyed the possibility of

using a standardized cloud architecture for interoperability [96]. Furthermore, Mobile cloud Computing (MCC) has been introduced to answer the need for easier interoperability, portability, and integration among heterogeneous platforms [97–100]. Finally, beyond the theoretical research on the use of cloud computing, many companies are providing cloud services at different scales and with a plethora of services; Amazon Web Services (AWS) [108], Microsoft Azure (Cloud and IoT Edge) [109], Google Cloud [110], Alibaba Cloud [111], IBM Cloud [112], Oracle Cloud [113], VMware Cloud [114] and many others. Because of the limitation in the cloud, Edge (also referred to as Fog) and mobile edge computing have been introduced [101–106] and there has been an effort by the OpenFog consortium to standardize the architecture of the Edge computing platform [107]. Various works have been proposed to deal with resource scheduling and and workload allocation in the edge and cloud [118–120]

These works have been great in handling the overwhelming amounts of data in terms of processing and storage; however, all these solutions target networks with specific prerequisites: the existence of these cloud/edge infrastructures in a nearby area, or the ability to access them through the Internet or some form of connection. Unfortunately, these conditions cannot be met in many applications where either there are no broadband infrastructures or the transfer of data is very costly and is only available at very low data rates. Besides the cost and infrastructures, sending data to the cloud/edge for processing takes time and hence hinders the ability to take immediate action. Furthermore, these computing paradigms are designed to meet the needs of a mobile user; therefore, their flexibility is bound by that of the users.

Today, data is as important as its source, and actions are directed by our ability to infer actionable information from the data at the source. Therefore, we pro-

pose a novel framework, which enables users to collect data using a low-latency routing algorithm, infer near-real-time actionable information and communicate necessary changes in a highly dynamic heterogeneous wireless networks, with a focus on agricultural MD-IoT networks. The framework features a low-latency routing algorithm for highly dynamic heterogeneous wireless networks with an integrated edge computing platform for real-time data processing.

5.2.2 Age of Information

Age of Information (AoI) is the measure of data freshness. It is a means to constrain the delay of receiving data and its usability. AoI was proposed in in the early nineties [121] but did not get much attention until it was used in [122] to measure the freshness and timely accuracy of remote resources' status updates. Kaul et al. [122] defined AoI (a.k.a. data freshness) as:

$$\Delta_\tau = \frac{1}{\tau}\int_0^\tau \Delta(t)dt \tag{5.1}$$

where τ is observation time interval and $\Delta(t)$ is the variation of age that increases over time until it is received by its intended destination. Age of Information is also written in simpler terms as the difference between the time at which it is received and its creation time.

Following its introduction, various age functions have been proposed [123–128] to cater for different application. For example, one could apply a penalty function in which the value increases with time indicating the staleness of the data or the opposite of that indicating the utility of data (deceasing with time) [123]. Data freshness is used in concurrence with other network measures such as energy saving [129]. Marbukh [130] proposed a mechanism to manage AoI in chal-

lenged Networks taking into consideration the effect of intermittent connections and varying delays on the freshness of data in the network. Scheduling and traffic management in wireless heterogeneous and real-time traffic have also been studied under the constraint of data freshness [131–133].

5.3 Network Description

5.3.1 Mobile, Delay-Sensitive Sensor-Actuator Network

In this Chapter, we study networks composed of two types of contacts that are prevalent in highly-dynamic networks: predicted and discovered. Our aim is to design a **network model** for **low-latency routing** using mobile and fixed nodes that have varying levels of storage, communication and processing capabilities. The routing model is then developed into a routing algorithm that enhances network efficiency in terms of end-to-end delay along with other criteria identified based on the application. This routing algorithm is used to assist the collection and aggregation of data, that is transferred to the edge computing platform that we developed for this kind of applications. The **edge computing platform** will provide services to the network, allowing near-real time creation of actionable information. We use an Ag-IoT application to emulated the proposed model and algorithm as it constitutes an important share in the IoT market and, as illustrated in the example that follows, it can greatly benefit from low-latency

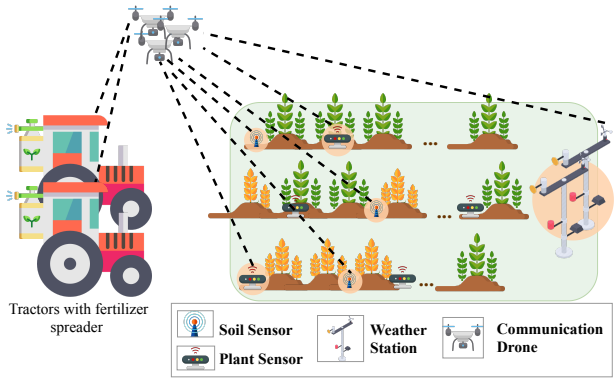


Figure 5.3: Scenario: Tractor with fertilizer sprayer.

routing and real-time data processing.

Consider a large agricultural field where fertilizer needs to be applied as shown in Fig. 5.3. Fertilizer is very expensive and can either be beneficial to the plants or damaging if there is too much or too little of it. In order to properly determine the right quantity of fertilizer for each spot, the machine needs to collect data about the plant (e.g. how green/healthy it is), the soil (e.g. how much fertilizer is already there and what nutrients are in the soil), and the current/future weather (e.g. whether it will rain or not and the state of the wind). The fertilizing implement also needs to communicate with the tractor in order to coordinate the actions on the fly based on the information collected from various types of sensors. For example, there could be three types of sensors:

- Machine sensors: send data ([$\sim$20 messages each of $\sim$15 bytes of data / second $\rightarrow$ 300B/s $\sim$18 KB/min]) about the status of the machine and how the machine (e.g. tractor) coordinates with the Implements (fertilizer in this scenario).

- Plant sensors: collect data about the state of the plants, such as the height, the color, etc. It could be in the form of images (e.g. from tractors and UAVs) or plaintext data (e.g. from Normalized Difference Vegetation Index (NDVI) and spectrometers).

- Ground sensors: such as weather stations and soil sensors. They are usually fixed but can be mobile and collect "slow data," which means data about environmental phenomena that do not change very frequently.

Different sensors are from different vendors and collect data with various volume, velocity and variety. Data from different sources can be needed to accomplish a

single task such as spraying fertilizer.

The network depicted in Fig. 5.3 should be designed in a way that allows data to flow seamlessly from all types of sensors to processing nodes, enables the latter to process data in a near-real-time fashion, and facilitates the transmission of actionable information back to the tractor in order to spray the fertilizer in a precise and timely manner. Designing such a network requires working on the selection of relay nodes taking into consideration not only their varying mobility models, but also their capabilities, types of data to be collected and delivered, coverage areas, delay tolerance, energy saving modes, reliability requirements, etc. It also depends on the precision and efficiency of data processing. Data can be processed in-the-network, or it can be offloaded to an edge computing platform that has more compute power and storage. These considerations, will be studied as we investigate optimizations in heterogeneous wireless networks composed of Predicted and Discovered contacts from the classification depicted in Fig. 2.2 in Chapter 2.

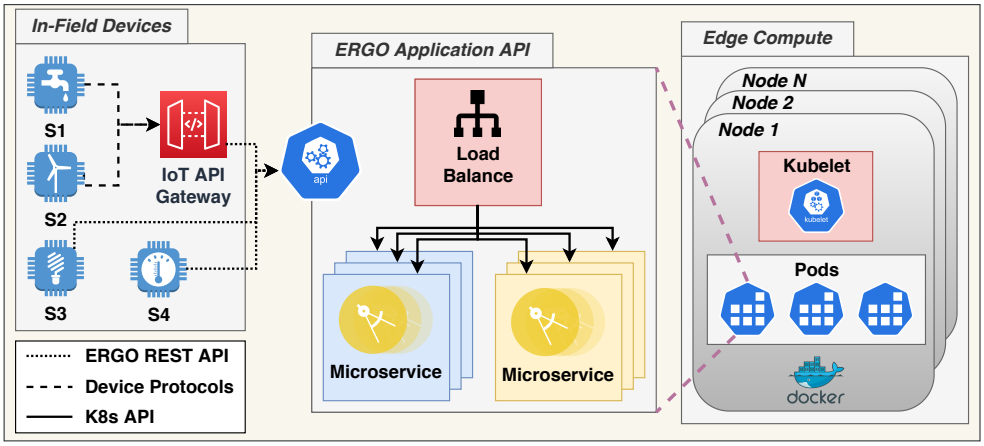


Figure 5.4: ERGO architecture.

5.3.2 ERGO, an Edge Architecture for Ag-IoT

We developed an edge computing architecture optimized for heterogeneous, resource-constrained IoT network for deployment in agricultural fields. These networks are typically infrastructureless and require near real-time processing of data collected in the field. Actuators receive actionable information from processing units that are expected to have more computing power than the mobile nodes that collect and relay data in the field. Along with collaborators [134], We proposed the edge architecture for Ag-IoT (ERGO) that features orchestration along with scalability to provide Ag-IoT application services using RESTful APIs. The architecture shown in Figure 5.4 is designed for networks with limited computational capabilities, bandwidth and energy resources. It is implemented as a cluster of computing nodes with dynamic management of application using a microservices architecture.

We use Kubernetes (K8s) [135] operations APIs to manage the cluster. Each node of the cluster hosts numerous pods. Each pod exposes application APIs implemented using the Flask web services gateway interface (WSGI) framework. Furthermore, each node runs a kubelet that configures and manages the pods' deployments. Finally, data collected by in-filed devices can be sent to ERGO either directly or through an IoT API aggregation gateway as shown in Figure 5.4. ERGO provides limited computing power; therefore, a load balancer service accepts jobs into a finite queue and dispatches them to different pods as resources become available. Once the system is full (i.e. all pods are busy and queue is full), jobs are dropped.

5.4 Network Model Specification

The network is defined as the graph $G_T = (\mathcal{V},\ \mathcal{E}_T,\ \mathcal{N})$ with the set of vertices $\mathcal{V}$ (i.e. sensor nodes and relay nodes), the set of temporal edge (also referred to as links) $\mathcal{E}_T$ and the set of processing nodes $\mathcal{N}$. The network runs multiple applications. The j^{th} application denoted by $\alpha_j \in \mathcal{A}$, is an application that uses the sensed data to compute and report actionable information to actuators in the network. $\mathcal{A} = [\alpha_1, \alpha_2, ..., \alpha_x]$ is the set of x applications that use the edge compute platform ERGO. Every application has particular traffic characteristics (i.e. text, video, imaging, etc), which we denote by traffic classes β_j. Each class with index j corresponds corresponds to the j^{th} application and defines its corresponding request size distribution θ_j. Requests arrive from each application with probability π_{α_j}.

5.4.1 Definitions and General Notations

Time is split into equal intervals $t_i, i \geq 0$ of length δt. In the model, the notation $||M||_1 = (\sum_{m \in M} |m|^p)^{\frac{1}{p}}$ is used to denote the p-norm of a matrix with $p = 1$. Because all the elements of the matrices used in the model are positive, the p-norm with $p = 1$ returns the sum over all the elements of the matrix. We also use the notations $(M)_i^r$ and $(M)_j^c$ to the denote the i^{th} row and the j^{th} column of matrix M respectively. We then define the generic function $\psi_\cdot(A, B, b) = \frac{||A \bullet (B)_b^c||_1}{\delta t}$ it is assumed that B's number of rows is equal to A's number of columns. $\psi_\cdot(A, B, b)$ is used later to compute the arrival rate at processing and relay nodes (differentiated using subscripts).

Objective function:

$$min\ \mathcal{D}_p\left(X_{t_i}\right) + \mathcal{D}_c\left(X_{t_i}\right) + \mathcal{P}\left(X_{t_i}\right) \tag{5.2}$$

X_{t_i} is the matrix mapping requests to paths at time t_i. If entry x_{ab} equals 1 means request a is to be sent through path b; the value is 0 otherwise. In this optimization problem, the objective is to minimize the path delay, $\mathcal{D}_p\left(X_{t_i}\right)$, and the processing delay, $\mathcal{D}_c\left(X_{t_i}\right)$, for all requests while enhancing the network throughput by applying the penalty function, $\mathcal{P}\left(X_{t_i}\right)$, to the unrouted requests. These three functions are defined as follows.

- **Path delay** is computed as the sum of the queuing delay and transmission delay over each path:

$$\mathcal{D}_p\left(X_{t_i}\right) = X_{t_i} W_p^*(X_{t_i}) + S_{t_i} X_{t_i} T_{t_i}^* \tag{5.3}$$

  where $W_p^*(X_{t_i})$ is defined in Sec. 5.4.2.4 Eq. (5.33) as a function of X_{t_i} that return a $P \times 1$ vector of the queuing delay of each path. Likewise, the expected transmission delay of all paths is represented by the $P \times 1$ vector, $T_{t_i}^*$, that is defined in Sec. 5.4.2.3 Eq. (5.28). S_{t_i} is a $Q \times Q$ diagonal matrix with the sizes of requests in bits.

- **Processing delay** is limited to the queuing delay at the processing node as we assume the nodes have enough capacity to process data is a negligible amount of time. It is however easy to add the processing delay to this function.

$$\mathcal{D}_c\left(X_{t_i}\right) = X_{t_i} W_n^*(X_{t_i}) \tag{5.4}$$

$W_n^*(X_{t_i})$ is defined in Sec. 5.4.2.1 Eq. (5.16) as a function of X_{t_i} that return a $P \times 1$ vector of the queuing delay of processing delay at the end of the each path.

- $\boldsymbol{\mathcal{P}(X_{t_i})}$ is a static penalty function that applies constant penalty for not routing a request using a large integer κ:

$$\mathcal{P}(X_{t_i}) = \sum_{a=0}^{Q} \left(1 - \sum_{x \in (X_{t_i})_a^r} x\right) \times \kappa \tag{5.5}$$

Subject to:

Processing Capacity – The sum of requested processing cycles does not exceed the processing power of the node data is sent to

$$B_{pr}(X_{t_i}, P_N) \leq B_{pr,ths} \tag{5.6}$$

Age of Information – The time to get to the processing node should be less than the age of the information transmitted through the request.

$$C_{\Delta}(X_{t_i}, D) \leq \Delta_{ths} \tag{5.7}$$

Energy – The sum of the energy spent on each request at node v should not exceed the total energy available at the node

$$C_E(E, S, X_{t_i}, P_V) \leq Y_t^V \tag{5.8}$$

Blocking probability – given demand intensity, what is the routing solution such that the blocking probability of each link/relay node does not exceed a blocking

threshold

$$B_L\left(S, X_{t_i}, P_V\right) \leq B_{L,ths} \tag{5.9}$$

Single Path Assignment – every request is assigned at most one path

$$\sum_{x \in (X_{t_i})_q^r} x \leq 1, \ \forall q \in [0, Q] \tag{5.10}$$

The parameters are listed in Table 5.1.

Table 5.1: Low Latency Routing Model parameters

Parameter	Description	Dimension
Q	total number of requests in time slot t_i	–
P	total number of paths in the network	–
N	number of processing nodes (i.e. $\|\mathcal{N}\|$)	–
V	number of network nodes including sensors and relay nodes (i.e. $\|\mathcal{V}\|$)	–
S	Matrix of all request sizes	$1 \times Q$
$B_{pr,ths}$	processing nodes blocking probability threshold	$1 \times N$
Δ_{ths}	Data freshness threshold	$1 \times Q$
Y_t^V	Energy available at each network node	$1 \times V$
$B_{L,ths}$	Link blocking threshold	$1 \times L$
p_L	Set of links in path p	–
P_L	$P_L(p,l) = 1$ if $l \in p_L$ and 0 otherwise	$P \times L$
P_N	$P_N(p,n) = 1$ if path p's final destination is node n and 0 otherwise	$P \times N$
P_V	$P_V(p,l) = 1$ if p traverses node v and 0 otherwise	$P \times V$
κ	Large penalty constant	–

5.4.2 Network Model

Model constraints are defined in this section.

5.4.2.1 Processing Nodes Blocking Probability and Waiting Time

There are N processing nodes in the network that operate independently and can be set up in different location. Each processing node, n, is assumed to run c_n pods

as described in Section 5.3.2.

Each processing node can be modeled as a c-server loss system, $M/M/c/K$, queuing system [136, Section 11.6], with $c = c_n$ pods and a queue of length $q_m^n = K - c_n$. The system follows the Poisson arrival process with a mean arrival rate λ_k when there are k jobs in the system and 0 when $k > K$. The service time is node-dependent and follows an exponential distribution with mean service rate $\mu_{n,k}$ when the population of the system is k. The arrival and process rates are defined as follows:

$$\lambda_k = \begin{cases} \lambda, & 0 \leq k < K \\ 0, & k \geq K \end{cases} \tag{5.11}$$

$$\mu_{n,k} = \begin{cases} k\mu_n, & 0 \leq k < c_n \\ c_n\mu_n, & c_n \leq k \leq K \end{cases} \tag{5.12}$$

The steady state equation for node n is given by:

$$\pi_{n,k} = \pi_{n,0} \prod_{i=1}^{k} \frac{\lambda_{i-1}}{\mu_{n,i}} \tag{5.13}$$

Using rates in Eq. (5.11) and (5.12), $\pi_{n,k}$ is written as shown in Eq. (5.14).

$$\pi_{n,k} = \begin{cases} \frac{1}{k!} \left(\frac{\lambda}{\mu_n}\right)^k \pi_{n,0}, & 0 \leq k < c_n \\ \frac{1}{c_n^{k-c_n} c_n!} \left(\frac{\lambda}{\mu_n}\right)^k \pi_{n,0}, & c_n \leq k \leq K \end{cases} \tag{5.14}$$

We use the function $\psi_n(X_{t_i}, P_N, n)$, defined in Sec. 5.4.1, to compute the arrival rate λ based on the expected traffic routed to node n (using X_{t_i} and P_N). For simplicity, we refer to it as ψ_n. The average waiting time in node n's queue is

given by $W_{n,t_i}(X_{t_i}, P_N)$ (referred to as W_{n,t_i}):

$$W_{n,t_i} = \frac{\sum_{k=c_n}^{K}(k - c_n)\pi^*_{n,k}}{\psi_n(1 - \pi^*_{n,K})} \tag{5.15}$$

$\pi^*_{n,K}$ is the value from Eq. (5.14) with $\lambda = \psi_n$. With that we define $W^*_n(X_{t_i})$ used in Eq. (5.4) as follows:

$$W^*_n(X_{t_i}) = P_N. \begin{bmatrix} W_{1,t_i} \\ \vdots \\ W_{n,t_i} \end{bmatrix} \tag{5.16}$$

Finally, the blocking probability is the probability that all the servers are busy and the queue is full, that is $\pi_{n,K}$, which can be obtained using Eq. (5.14) and the value of $\pi_{n,0}$ written as:

$$\pi_{n,0} = \left[\sum_{i=0}^{c_n - 1} \frac{1}{i!} \left(\frac{\lambda}{\mu_n} \right)^i + \sum_{i=c_n}^{K} \frac{1}{c_n^{i-c_n} c_n!} \left(\frac{\lambda}{\mu_n} \right)^i \right]^{-1} \tag{5.17}$$

The blocking probability in Eq. (5.14) is used Eq. (5.6) with the following changes. We rewrite Eq. (5.14) to compute the expected blocking probability, $\pi^*_{n,K}$, by substituting the time-variant ψ_n for λ as follows:

$$\pi^*_{n,K}(X_{t_i}, P_N) = \frac{1}{c_n^{K-c_n} c_n!} \left(\frac{\psi_n}{\mu_n} \right)^K \pi^*_{n,0} \tag{5.18}$$

In Eq. (5.18), $\pi^*_{n,0}$ is the expected value computed using ψ_n and the result from Eq. (5.17). With this, the constraint in Eq. (5.6) is as follows:

$$\pi^*_{n,K}(X_{t_i}, P_N) \leq \gamma_{pr,t_i} \ , \ \forall n \in \mathcal{N} \tag{5.19}$$

where $B_{pr,ths} = \gamma_{pr,t_i}$ is the blocking threshold that should not be exceeded and

$B_{pr}(X, P_N) = \pi^*_{n,K}(X_{t_i}, P_N)$.

5.4.2.2 Age of Information

Age of Information, denoted by Δ_{t_i} at time t_i, measures the freshness when data is received at the destination at time τ and is given as [122, 123, 128]:

$$\Delta_{t_i} = \tau - \mathcal{C}_{u,t_i} \tag{5.20}$$

Where $\mathcal{C}_{u,t_i}$ is the data creation time by node u. In the studied network, that would be equivalent to the time at which data is acquired by sensor u. The path from source to destination is modeled using a single-server queuing system with independent and identically distributed (i.i.d.) service times. The queue follows FIFO policy; however, data is not served following its creation time, but the time at which it arrives to the system, denoted by a_{u,t_i}. The arrival time to the system satisfies the following inequality: $\mathcal{C}_{u,t_i} \leq a_{u,t_i}$. Given the significance of data freshness in delay sensitive networks, we define an AoI penalty function, $\omega(\Delta_{t_i})$, and a data freshness threshold, $\Delta_{ths} = \gamma_{\Delta,t_i}$, to bind the path delay, which is the main source of data staleness.

The estimated time to receive data is dependent on the path delay and is written as:

$$\tau^*(X_{t_i}) = a_{u,t_i} + \mathcal{D}_p(X_{t_i}) \tag{5.21}$$

where $\mathcal{D}_p(X_{t_i})$ is the path delay of path p and is defined in Eq. (5.3), and $\tau^*(X_{t_i})$ will be referred to as τ^*. The corresponding estimated AoI is:

$$\Delta^*_{t_i} = \tau^* - \mathcal{C}_{u,t_i} \tag{5.22}$$

With that, the constraint for data freshness (see Eq. (5.7)) is defined in Eq. (5.23)

$$\omega(\Delta^*_{t_i}) \leq \gamma_{\Delta,t_i} \tag{5.23}$$

where $C_\Delta(X, D) = \omega(\Delta^*_{t_i})$ The penalty function is non-decreasing since penalty on data staleness increases with time reducing the interest in the data. For networks studied in this work, the penalty function can apply a time-variant application(α_j)-dependent cost, ν_{α_j,t_i}, to the estimated data freshness, $\Delta^*_{t_i}$.

5.4.2.3 Node Energy Availability

Energy consumption of node u at time t_i is computed using the residual energy at time t_{i-1} and the energy to be consumed by the requests routed through the node. For that we use the formula in Eq. (5.24).

$$\mathcal{Y}_{u,t_i} = \begin{cases} \mathcal{Y}_{u,t_{i-1}} - \sum_{q \in \mathcal{Q}_v} E^*_{q,u,t_{i-1}}, & i > 0 \\ \mathcal{Y}_{u,max}, & i = 0 \end{cases} \tag{5.24}$$

where $\mathcal{Y}_{u,max}$ denotes the maximum available energy of node u. The achievable rate at time t given by the Shannon-Hartley formula as $r_{u,t} = \mathcal{B}\, log_2\left(1 + \frac{g_{u,t} w_{u,t}}{\eta}\right)$ is used to compute the value of $E^*_{q,t_{i-1}}$ as specified in Eq. (5.25):

$$E^*_{q,u,t_{i-1}}(q_s) = w_{u,t_i} . \frac{q_s}{r_{u,t_{i-1}}} \tag{5.25}$$

In Shannon-Hartley formula, $\mathcal{B}$ is the system bandwidth, g_{u,t_i} is the channel power gain between node u and the other end of the link, w_{u,t_i} is the transmit power and η is the noise power at the receiver. q_s denotes the request size in bits.

As such, the constraint $C_E(S, X, P_V)$ in Eq. (5.8) is written using the aforemen-

tioned definition as follows:

$$\sum_{q=1}^{Q} E^*_{q,u,t_i}(q_s)||(X_{t_i})^r_q, P_V||_1 \leq \mathcal{Y}_{u,t_i}, \ \forall u \in \mathcal{V} \tag{5.26}$$

Using the same Shannon-Hartley formula we can compute the expected transmission delay over path p as:

$$T^*_{p,t_i} = \sum_{l \in p_L} \frac{1}{r_{l(u),t_{i-1}}} \tag{5.27}$$

Where $l(u)$ is the source node of link l, and p_L is the set of links in path p. Finally, we define the $P \times 1$ matrix of path transmission delays in Eq. (5.3) as follows:

$$T^*_{t_i} = \begin{bmatrix} T^*_{1,t_i} \\ \vdots \\ T^*_{p,t_i} \end{bmatrix} \tag{5.28}$$

5.4.2.4 Link Blocking Probability

Relay nodes in the networks are assumed to have limited space for the forwarding buffer; therefore, each relay node u can be modeled as an $M/M/1/K_u$ queuing system with a queue size K_u, a Poisson arrival process (i.e. arrivals are independent and identically distributed) and a service rate that follows an exponential distribution with mean service rate $\mu_{u,tx}$. When the node is saturated, i.e. the queue is full with K_u packets in the system, newly arriving packets are dropped. The arrival rate, is hence equal to σ when there are at most K_u packets in the node and 0 otherwise. The probability that arriving packets are dropped is equivalent to the probability that there are k_u packets, π_{K_u}.

We define $\rho_{u,tx} = \frac{\sigma}{\mu_{u,tx}}$ and assume $\rho_{u,tx} \neq 1$ (i.e. $\sigma \neq \mu_{u,tx}$). The probability

π_{K_u} is given as [136, Section 11.5]:

$$\pi_{K_u} = \frac{(1-\rho_{u,tx})\,(\rho_{u,tx})^{K_u}}{1-(\rho_{u,tx})^{K_u+1}} \tag{5.29}$$

Here also we use the function $\psi_s(X, P_V, u)$ (defined in Sec. 5.4.1 and referred to as ψ_s for convenience) that computes the arrival rate at node u based on the requests routed through node u. The formula given in Eq. (5.29) is then used to compute the estimated blocking probability at node u and time t_i:

$$\pi^*_{K_u,t_i}\left(X_{t_i}, P_V\right) = \frac{\left(1-\rho^*_{u,tx}(\psi_s)\right)\left(\rho^*_{u,tx}(\psi_s)\right)^{K_u}}{1-\left(\rho^*_{u,tx}(\psi_s)\right)^{K_u+1}} \tag{5.30}$$

where $\rho^*_{u,tx}(\psi_s) = \frac{\psi_s(X_{t_i},P_V)}{\mu_{u,tx}}$ is the value of $\rho_{u,tx}$ computed using the arrival rate returned by the function ψ_s. We can then rewrite Eq. (5.9) as follows by substituting $B_L\left(X_{t_i}, P_V\right)$:

$$\pi^*_{K_u,t_i}\left(X_{t_i}, P_V\right) \leq B_{L,ths},\ \forall u \in \mathcal{V} \tag{5.31}$$

In addition to the blocking probability, we can compute the expected average queue delay of path p as:

$$W^*_{p,t_i} = \sum_{v \in p_V} \frac{1}{\psi_s\left(1-\pi^*_{K_v,t_i}\right)} \mathcal{L}^*_{v,t_i} \tag{5.32}$$

where p_V is the set of nodes in path p and $\mathcal{L}^*_{v,t_i}$ is the expected average number of customers in the queue and can be computed using the $M/M/1/K$ performance measures equations such as ones presented in [136, Section 11.5]. In the latter, we

substitute ρ by $\rho^*_{u,tx}(\psi_s)$. Lastly, $W^*_p(X_{t_i})$ (in Eq. (5.3)) is written as:

$$W^*_p(X_{t_i}) = \begin{bmatrix} W^*_{1,t_i} \\ \vdots \\ W^*_{p,t_i} \end{bmatrix} \tag{5.33}$$

5.4.3 Routing dependent variables

Depending on the routes from each sensor to the processing nodes, parameters D, R, P_L, P_N, and P_V take different values.

The optimization problem is formulated under assumption that each sensor knows the full path to each processing node, and the routes do not change for the whole network lifetime since the nodes are fixed.

5.5 Model solving

The objective function in Eq. (5.2) can be simplified as:

$$\begin{aligned} min\ f(X_{t_i}) = & \\ & \sum_{q=1}^{Q} \sum_{p=1}^{P} x_{q,p} w^*_{\overline{p}}(x_{1,p}, \ldots, x_{q,p}) \\ & + \sum_{q=1}^{Q} q_s \sum_{p=1}^{P} x_{q,p} T^*(p) \\ & + \sum_{n=1}^{N} \sum_{q=1}^{Q} \sum_{p=1}^{P} x_{q,p} w^*_{\overline{n}}(x_{1,p}, \ldots, x_{q,p}) \times y_{n,p} \\ & + \sum_{a=0}^{Q} \left(1 - \sum_{x \in (X_{t_i})^r_a} x \right) \times \kappa \end{aligned} \tag{5.34}$$

where $x_{q,p}$ is the element (q, p) of the matrix X_{t_i}, we drop the time subscript for convenience. $w^*_{\overline{p}}$ is the path queuing time as function of column p of the matrix X_{t_i} as described in Sec. 5.4.2.4. q_s is the element of S representing the size in bits of request q and $T * (p)$ is as described in Sec. 5.4.2.3. $w^*_{\overline{n}}$ is the queuing delay at processing node n and is also a function of column p of the matrix X_{t_i} (see Sec. 5.4.2.1), and $y_{n,p}$ is element (n, p) of the matrix P_N indicating whether or not path p goes to node n.

Subject to:

$$\pi^*_{n,K}(X_{t_i}, P_N) \leq \gamma_{pr,t_i}\ ,\ \forall n \in \mathcal{N} \tag{5.35}$$

$$\omega(\Delta^*_{t_i}) \leq \gamma_{\Delta,t_i} \tag{5.36}$$

$$\sum_{q=1}^{Q} E^*_{q,u,t_i} ||(X_{t_i})^r_q, P_V||_1 \leq \mathcal{Y}_{u,t_i},\ \forall u \in \mathcal{V} \tag{5.37}$$

$$\pi^*_{K_u,t_i}(X_{t_i}, P_V) \leq B_{L,ths},\ \forall u \in \mathcal{V} \tag{5.38}$$

$$\sum_{x \in (X_{t_i})^r_q} x \leq 1,\ \forall q \in [0, Q] \tag{5.39}$$

Notice that all four terms of the objective function and all four constraints are written in terms of information about other nodes' requests in the network (i.e. every i^{th} row of the matrix $(X_{t_i})^r_i$). This optimization problem is an integer quadratic programming (IQP) with Quadratic constraints; which are proved to be NP-hard [137]. Furthermore, n a highly dynamic network that requires real-time data processing, sharing routing information between all nodes would be very costly and unfeasible. Each node in the network will receive routing information from $V - 1$ nodes every δt time units; however, to generate this routing information, each node will also need to know decision of the other nodes. A centralized solution that computes routes and sends them to each sensor is also not feasible

in a resource constrained network with real-time applications.

In order to keep that problem distributed, we identify the main functions that require external information to be:

- $\psi_n(X_{t_i}, P_N, n)$ (used in Constraint (5.35)) returns the expected arrival rate at node processing node n during time slot t_i.
- $\psi_s(X_{t_i}, P_V, u)$ (used in Constraints (5.37) and (5.38)) computes the expected arrival rate at each relay node u during time slot t_i.
- $E^*_{q,u,t_i}||(X_{t_i})^r_q, P_V||_1$ in Eq. (5.37) which represents the average number of bits transferred through node u during time slot t_i. This can also be written as function of $\psi_s(X_{t_i}, P_V, u)$ and the mean requests size.

We introduce three algorithms that are used to predict estimates of each value, which will allow us to simplify the model by reducing the dimension of the decision variable matrix from $Q \times P$ to $1 \times P$ and run it in a distributed fashion. The simplified model results in sub-optimal routing solutions whose optimality depends on the precision of the predictions.

5.5.1 Arrival Rate $\psi_s(X_{t_i}, P_V, u)$

We propose the COllaborative weighTed Average for relay nodes ($COTA^u$) algorithm to compute the arrival rate collaboratively by clusters of nodes in the network. Each node u receive information from its immediate neighbors about the their observed average arrival rate across different time slots. Node v keeps a running average of its observed arrival rate, $\overline{\lambda^v_{t_{i+1}}}$, which it computes using the

number of observed arrivals at time t_{i+1} (a.k.a. $NOR^v_{t_{i+1}}$) as:

$$\overline{\lambda^v_{t_{i+1}}} = \frac{\left(\overline{\lambda^v_{t_i}} \times i\right) + NOR^v_{t_{i+1}}}{i+1} \tag{5.40}$$

The observed arrival rate computed in Eq. (5.40) by each neighbor, v, along with the density of nodes around it (in a certain range) is piggybacked in communication packets every, $k \times \delta t$ set for efficiency. Using $COTA^u$ algorithm detailed in Algorithm 7, node u computes the weighted average of the arrival rate at each relay node based on local information of the network (i.e. the density, arrival rate pairs collected from neighboring nodes). The set of neighbors of node u is denoted by E_u. $COTA^u$ assigns a weight to each node $v \in E_u$ that uses the exponential decay function of the distance between v and a, the target relay node. The decay constant γ is a positive value, and intuitively, the larger it is, the smaller the weight assigned to distant nodes.

Algorithm 7 $COTA^u$ Algorithm

Input:
    Neighbor set E_u .
    Relay node a;
Output:
    $\overline{\lambda^a_{t_{i+1}}}$.

for all $v \in E_u$ **do**
    $w_v \leftarrow w_0 e^{-\gamma d(v,a)}$; ▷ $w_0 = 1$
$\overline{\lambda^a_{t_{i+1}}} \leftarrow \sum_{v \in E_u} w_v \overline{\lambda^v_{t_i}} / \sum_{v \in E_u} w_v$;

5.5.2 Arrival Rate $\psi_n(X_{t_i}, P_N, n)$

The computation of arrival rates are processing nodes can be performed using the average traffic traversing the routes that lead to each one of them. That is, we

use P_N to compute the arrival rate at node $n \in \mathcal{N}$. We propose the COllaborative weighTed Average for processing nodes ($COTA^n$) and describe it in Algorithm 8. The algorithm is based on that premise that if relay node a's neighbor set E_a con-

Algorithm 8 $COTA^n$ Algorithm

Input:
    Processing node n;
    Neighbor set of node n E_n .
Output:
    $\overline{\lambda^n_{t_{i+1}}}$.

for all $a \in E_n$ **do**
  $c_{sum} \leftarrow 0$;
  for all $v \in E_a$ **do**
    if $v \in \mathcal{N}$ **then**
      $c_{sum} \leftarrow c_{sum} + c_v$;
  $w_{n,a} = \frac{c_n}{c_{sum}}$
$\overline{\lambda^n_{t_{i+1}}} \leftarrow \sum_{a \in E_n} w_{n,a} \overline{\lambda^a_{t_i}}$; ▷ Use $COTA^u$ to compute $\overline{\lambda^a_{t_i}}$

tains multiple processing nodes, than the traffic arriving at a would be distributed across all these processing nodes. We further assume that the portion of traffic to be sent to each processing node n is dependent on its processing capacity c_n, which we previously referred to as the number of servers (see Sec. 5.4.2.1).

5.5.3 Energy Consumption

The residual energy at relay node a at time t_{i+1} can be computed as a function of the average arrival rate, the average requests' size, and their associated probability. We propose the Residual Energy Estimates (R2E) algorithm, described in Algorithm 9, which uses Eq. (5.24) and Eq. (5.25) to compute the residual energy at node a.

Algorithm 9 *R2E* Algorithm

Input:
    Set of applications $\mathcal{A}$.
    Relay node a;
Output:
    $\overline{\lambda^{r}_{t_{i+1}}}$.

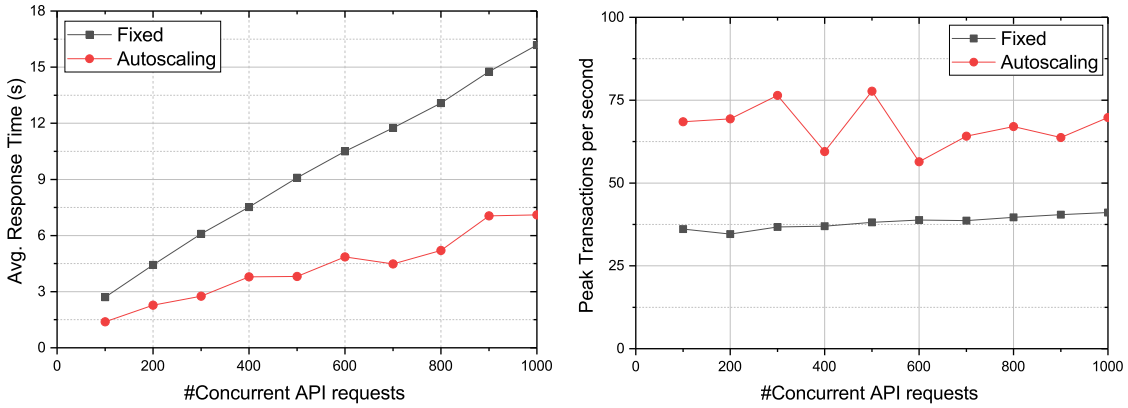
if $i = 0$ **then**
  $\overline{\mathcal{Y}_{a,t_i}} \leftarrow \mathcal{Y}_{u,max}$;
else
  $r_{a,t_{i-1}} \leftarrow \mathcal{B}\ log_2\left(1 + \frac{g_{a,t_{i-1}} w_{a,t_{i-1}}}{\eta}\right)$;
  $\overline{q_s} = \overline{\lambda^{a}_{t_{i-1}}} \sum_{\alpha_j \in \mathcal{A}} \left(\theta_j \times \pi_{\alpha_j}\right)$; ▷ Use $COTA^{u}$ to compute $\overline{\lambda^{a}_{t_{i-1}}}$
  $\overline{\mathcal{Y}_{a,t_i}} \leftarrow \overline{\mathcal{Y}_{u,t_{i-1}}} - w_{a,t_{i-1}} \cdot \frac{\overline{q_s}}{r_{a,t_{i-1}}}$;

5.5.4 The Simplified Model

The new problem that uses arrival rate and residual energy computation algorithms is an Integer Linear Programming (ILP) model defined, which is less accurate than the original model, but more efficient as we show later in the evaluation section. We define the new problem a follows.

$$min\ f(X_{p,t_i}) = \sum_{p=1}^{P} x_p \overline{w^{*}_{p^*}(x_p)} + \overline{q_{total}} \sum_{p=1}^{P} x_p \overline{T^{*}(p)} + \sum_{n=1}^{N} \sum_{p=1}^{P} x_{q,p} \overline{w^{*}_{n^*}(x_p)} \times \overline{y_{n,p}} \tag{5.41}$$

we use $\bar{\cdot}$ to denote the element of the objective function that are predicted using Algorithms 8, 7 and 9. The corresponding constraints are also simplified, so that instead of using matrix X_{t_i} to compute network parameters, the latter are provided using the aforementioned algorithms. In the next section (Sec. 5.6), we provide a detailed description of the implementation and corresponding evaluation.



(a) ERGO API response times.

(b) ERGO API throughput.

Figure 5.5: Performance evaluation of the Ag-IoT Application APIs.

5.6 Experimental Setup and Results

5.6.1 ERGO Performance Measurement

A prototype of the ERGO cluster was implemented using a 5-node Raspberry Pi 4 Model B cluster with quad-core ARM A72 64-bit SoC and 4GB DDR4-3200 SDRAM. Four of the nodes, each with 4 cores, are used for application deployment and the fifth is reserved for node control and management. The experimental data is derived from the scenario in Sec. 5.3.1, where we generate synthetic data that mimics real world values.Multiple pieces of information are sent to the cluster, where they are aggregated by time and used in the computation of the action that a fertilizing tractor should take. The performance is shown in Figures 5.5a and 5.5b. We run the experiments with and without auto-scaling to evaluate the scalability of ERGO, and as shown in the figures, the performance of ERGO is significantly enhanced when we use auto-scaling. That is, we are able to achieve lower response time (Fig. 5.5a) while increasing the cluster's throughput (fig. 5.5b), two metrics that are important in a low-latency, resource constrained environment.

5.6.2 The Low-Latency ILP Routing Evaluation

In order to evaluate the performance of the proposed routing algorithm and corresponding ILP model, we implement a MATLAB simulator. The simulator creates a pseudo-random network, similar to the one shown in Fig. 5.6. The network is composed of a number of processing nodes, sensor nodes and relay nodes. Traffic is generated following multiple types of applications similar to ones used in Sec. 5.6.1.

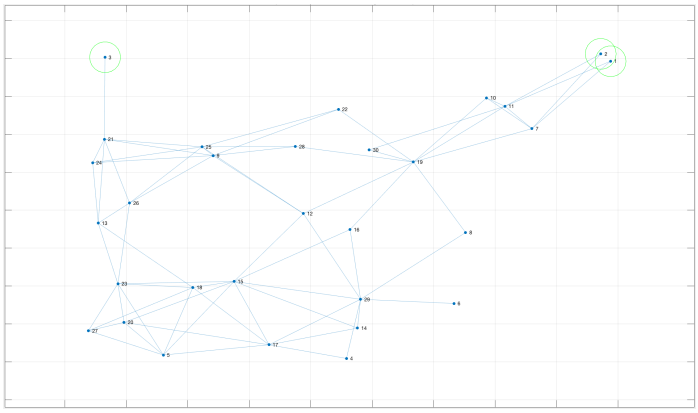
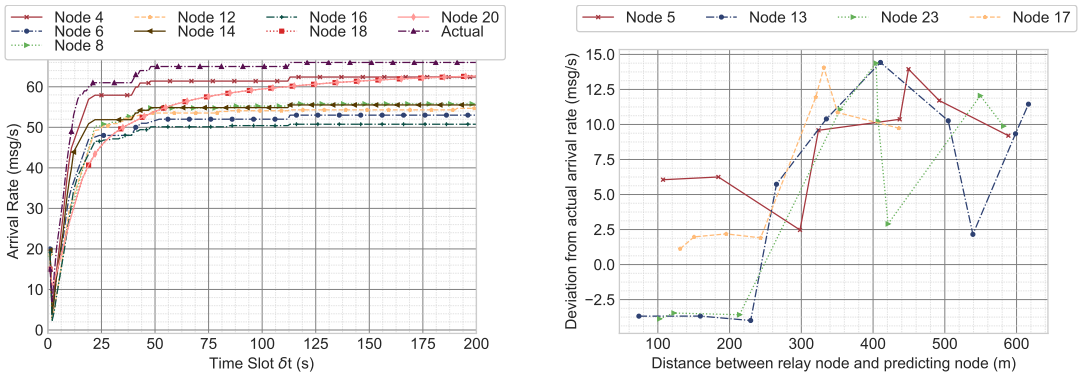


Figure 5.6: Example experimental network. Green circle indicates ERGO clusters.

ERGO's arrival and service Poisson processes were chosen based on values obtained through the evaluation of the cluster as described in the same section. Finally, we use MATLAB's *intlinOpt* method for ILP model solving. In every time slot, a number of messages are created by sensor nodes. Each sensor node predicts arrival rates at relays nodes along available paths using $COTA^u$, arrival rate at the processing nodes (destination of the each path) using $COTA^n$ and the residual energy at each relay node using R2E. The values are then used by the ILP solver and an optimal path is returned. Finally, the network is state is updated based on the node's routing decisions. We analyze the performance of the proposed algorithms by comparing the computed $\overline{\lambda^a_{t_{i+1}}}$, $\overline{\lambda^n_{t_{i+1}}}$, and $\overline{\mathcal{Y}_{a,t_i}}$ to the actual values of the simulation. Figure 5.7a shows sensor nodes' predicted arrival rate for relay node 5. The figure shows that $COTA^u$ is able to predict accurate values based on the neighboring node's arrival rates the their distance to the tar-



(a) Predicted arrival rate for relay node 5.

(b) Effect of distance on predicted arrival rate of different relay nodes.

Figure 5.7: Performance evaluation of $COTA^u$.

get node with an average magnitude of relative error equal to 13.8% and ranging between 3.9% and 22%. $COTA^u$, however, does not generate accurate predictions for nodes that are not used in the network (i.e. $\overline{\lambda^a_{t_{i+1}}} \approx 0$ because its prediction is based on local information. This is also shown by the results in Figure 5.7b, which depicts the effect of the distance between nodes on the value predicted. The farther the node is from a relay node, the higher the error in arrival rate prediction because of the exponential decay function used for weight computation.

Finally, Figure 5.8 shows the performance of the routing model in terms of message delivery to the processing nodes and dropping percentage at the latter. Our low-latency routing ILP formulation can deliver up to 85.6% of messages to processing nodes, and thanks to the blocking proba-

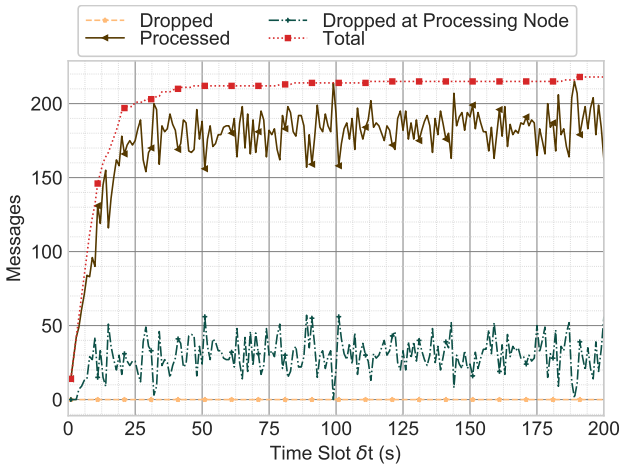


Figure 5.8: Traffic delivery and processing.

bility threshold constraint, only 14.3% were dropped at the processing node.

5.7 Conclusions and Future Work

In this chapter, we studied routing optimization and edge computing in delay-sensitive, resource constrained heterogeneous networks. We specifically considered the case of Agricultural-IoT networks as they constitute a prominent example of such networks. Ag-IoT networks are deployed in areas that have limited communication and compute infrastructure, or a complete lack thereof. This means that it is expensive and time consuming, if not impossible, to send data from sensors to remote cloud platforms. Furthermore, data collected by sensors is transient as monitored phenomena in the field can change rapidly. That being said, we proposed in this chapter a scalable edge compute platform and a network optimization model that finds the optimal path from sensor to processing nodes.

Our proposed ERGO, the edge architecture for Ag-IoT, is a cluster-based platform that utilizes Kubernetes to dynamically manage the nodes of the cluster. Each of these nodes can also dynamically deploy and remove pods (i.e. single processing units). ERGO's pods expose user application , which our implementation, are exposed using Flask web Services gateway interface. Field data is sent to ERGO through relay nodes, a process that we modeled as an IQP model. The model minimizes the delay in sending data to the edge platform while maximizing the delivery ratio. The objective function is subjected to three main constraints related to network blocking (i.e. traffic dropping), Age of Information and energy consumption at sensor and relay nodes. Because of the complexity of IQP models, we reduced our proposed model to a LIP model by proposing network metrics

prediction algorithms.

In order to evaluate our proposed routing model, we implemented and run network simulations on MATLAB. We generate pseudo-random networks that are composed of a specific number of processing clusters and multiple sensor and relay nodes. The generated traffic also follows numbers from real world Ag-IoT applications, such as the one described in Sec. 5.3.1. The simulation starts by finding the set of paths from sensor to edge compute clusters. It then uses MATLAB's the Integer Liner Programming function (intlinprog) to find the optimal path for each request. *intlinprog* bases the optimization on values generated using our proposed algorithms $COTA^u$, $COTA^n$ and $R2E$. The results show that the prediction are accurate for for networks where nodes are equally distributed across the field (i.e. each node is like to receive similar numbers of messages).

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

In this dissertation, we investigate the topic of routing optimization in heterogeneous wireless networks with a focus on space and mission-driven IoT environments. We started by providing a comprehensive mathematical classification for the contacts in heterogeneous wireless networks, defining four main categories of contacts: scheduled, discovered, predicted and continuous. Based on this classification, we developed models and routing algorithms for networks composed of scheduled contacts only, ones composed of both predicted and discovered, and finally networks with all these types of contacts.

Space Networks (a.k.a. Interplanetary Networks (IPN)), which we used as a use-case scenario of scheduled networks, are investigated first in this dissertation. Space networks are considered Delay Tolerant Networks (DTN) because they are deployed in a challenged environment where delay and link disruption are the norm and not the exception. We introduced the Modified Temporal Graph (MTG) model that allowed us to describe these networks in a near-real- time deterministic dynamic representation. Using the MTG, we proposed the first Mixed Integer Linear Programming (MILP) model for message routing and scheduling in space

networks, and developed two sliding window heuristics. N-Look Ahead Routing and Scheduling (N-LARS) algorithm that considers only one message sorting criterion, and Multi-Attribute Routing and Scheduling (MARS) algorithm that uses Multi-Attribute Decision Making to sort outgoing messages. Our experimental evaluation of the algorithms, using realistic networks, showed that they can enhance the network throughput and overall end-to-end delay significantly, compared to the currently used routing algorithm the Contact Graph Routing (CGR). Our proposed routing algorithms are also shown to be more scalable than CGR as it could route messages in relatively large networks for which CGR experiments could not be completed.

Working on networks on the other side of the spectrum, we investigated routing and optimization in delay-sensitive heterogeneous wireless networks in the form of Mission-Driven IoT. We took as example networks Agriculture IoT and disaster recovery networks. We have used the latter as a network that is composed of all four types of contacts. We proposed a Statistical Analysis (STAN) framework that we use to analyze contact traces as a time series. We then designed it to choose one of the available classical and deep learning forecast methods to generate a list of predicted contacts. The predicted future contact traces are then shared with our proposed Predictive Routing algorithm, PETRA. We evaluated this work using real traces from a fire department and showed that using statistical analysis and choosing the right forecast model can enhance the end-to-end delay and throughput of the network compared to one of the routing algorithms in the literature that uses CNN for contact forecasting.

Last but not least, we analyzed and designed a network model and architecture for predicted and discovered contacts in large dynamic heterogeneous wireless MD-IoT networks using Agricultural IoT (Ag-IoT) as use-case scenario.

In this work, we considered various aspects of heterogeneity present in an Ag-IoT network such as technology, devices, traffic, and quality requirements. These networks also require a near-real time processing of data in order to extract actionable information that can be used to maintain the farm and manage its resources. Considering all these factors, we implemented a low-latency ILP-based routing algorithm for such a highly dynamic network that we integrate with our proposed edge computing framework for data processing in resource-constrained networks. In this work, completed with collaborators from the school of Agricultural Sciences and Natural Resources, we use agricultural data collected in large fields to evaluate our proposed ERGO, model and algorithms. The simulation results show that ERGO is capable to enhancing the throughput and reducing the processing latency by auto-scaling depending on the jobs arrival rate. We also show that the proposed network parameters prediction algorithms provide the model with a good approximation to compute paths.

6.2 Future Directions

Our dissertation work on routing optimization in heterogeneous wireless networks with a focus on space and MD-IoT environments demonstrates the great benefits and importance of inherently embracing heterogeneity as a norm and not an exception in designing network architectures and algorithms. Though it can take many forms, heterogeneity should be encouraged rather than eliminated because it brings innovation, scalability and robustness to the network. In our work, we have used various application domains to demonstrate and evaluate our proposed frameworks and algorithms, such as space networks, disaster recovery MD-IoT networks, and agricultural IoT networks.

Our contributions can be extended to many other heterogeneous wireless networks such as intelligent transportation systems, vehicular networks, smart cities, smart buildings, smart manufacturing, etc. Though they have different applications and requirements, all these examples are similar in that they are built on an underlying heterogeneous wireless network that in many cases necessitates the integration of near-real time data processing. With our contributions, we have proved that heterogeneity can be fostered for future efficient and scalable network architectures and provided details of frameworks and algorithms that can be extended for a variety of applications. In-filed implementations using new communication technologies, such as millimeter wave, can add more value to our understanding of the network dynamics in light of heterogeneity. Finally, the performance of our proposed low-latency routing model could be further enhanced by considering an algorithm that uses techniques such as Machine Learning or Deep Learning to predict network parameters.

Bibliography

[1] H. El-Damhougy and K. Jerelt, "Interplanetary communications network, interplanetary communications network backbone and method of managing interplanetary communications network ," Application US20 080 151 811 A1, 06 26, 2008, US Patent App. 11/613,839.

[2] V. Chan, J. Chapin, P. Elson, D. Fisher, V. Frost, K. Jones, and G. Miller, "Future heterogeneous networks," USA, Tech. Rep. https://dl.acm.org/doi/book/10.5555/2581999

[3] S. El Alaoui, S. Palusa, and B. Ramamurthy, "The interplanetary internet implemented on the geni testbed," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[4] S. El Alaoui and B. Ramamurthy, "Routing optimization for DTN-based space networks using a temporal graph model," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[5] S. El Alaoui and B. Ramamurthy, "N-look ahead routing and scheduling (N-LARS) for DTN space networks," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[6] ——, "EAODR: A novel routing algorithm based on the Modified Temporal Graph network model for DTN-based Interplanetary Networks," *Computer Networks*, vol. 129, Part 1, pp. 129 – 141, December 2017.

[7] ——, "MARS: A Multi-Attribute Routing and Scheduling Algorithm for DTN Interplanetary Networks," *IEEE/ACM Transactions on Networking*, in press.

[8] S. El Alaoui and B. Ramamurthy, "stan+petra: A statistical analysis aided routing algorithm for qos in mission-driven iot networks," in *2019 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, December 2019, pp. 1–6.

[9] European Space Agency (ESA). (2015, April) METERON. [Online]. Available: http://www.esa.int/Our_Activities/Human_Spaceflight/International_Space_Station/Meteron

[10] B. Dunbar. (2017, August) Journey to Mars Overview. Accessed Oct 16, 2019. [Online]. Available: https://www.nasa.gov/content/journey-to-mars-overview

[11] The Consultative Committee for Space Data Systems (CCSDS), "Cislunar space internetworking – architecture," December 2006, Draft Informational Report, CCSDS 730.1-G-0 .

[12] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," April 2007, IETF Request for Comments, RFC 4838. [Online]. Available: https://tools.ietf.org/html/rfc4838

[13] J. Morgenroth, W.-B. Pöttner, S. Schildt, L. Wolf, "12 - Performance issues and design choices in delay-tolerant network (DTN) algorithms and protocols ," in *Advances in Delay-Tolerant Networks (DTNs)*, J. Rodrigues, Ed. Oxford: Woodhead Publishing, 2015, pp. 225 – 250.

[14] A. Vasilakos, Y. Zhang, and T. Spyropoulos, *Delay Tolerant Networks: Protocols and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2011.

[15] A. I. Lysenko and S. V. Valuiskyi, "Calculation of MANET subscibers connectivity time," in *2011 21st International Crimean Conference "Microwave Telecommunication Technology"*, Sept 2011, pp. 361–362.

[16] Jet Propulsion Laboratory. (2005?) Technologies of Broad Benefit: Propulsion. [Online]. Available: http://mars.nasa.gov/mer/technology/bb_propulsion.html

[17] P. Romano, P. Schrotter, O. Koudelka, and M. Wittig, "Developments towards an interplanetary internet," in *2009 International Workshop on Satellite and Space Communications*, Sept 2009, pp. 310–314.

[18] European Space Agency. Our Missions. [Online]. Available: http://www.esa.int/ESA/Our_Missions

[19] R. Andersen. (2014, September) The Elon Musk Interview on Mars Colonization. [Online]. Available: http://aeon.co/magazine/technology/the-elon-musk-interview-on-mars/

[20] Jet Propulsion Laboratory. Future Missions. [Online]. Available: https://www.jpl.nasa.gov/missions/?type=future

[21] The State Council Information Office of the People's Republic of China. (2016, December) China's Space Activities in 2016. [Online]. Available: http://www.scio.gov.cn/zxbd/wz/Document/1537091/1537091.htm

[22] T. Greicius. (2019, August) Nasa's mars helicopter attached to mars 2020 rover. [Online]. Available: https://www.nasa.gov/feature/jpl/nasas-mars-helicopter-attached-to-mars-2020-rover

[23] NASA Space Science Data Coordinated Archive. (2018) Chang'e 4. [Online]. Available: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2018-103A

[24] N. Bezirgiannidis, C. Caini, and V. Tsaoussidis, "Analysis of contact graph routing enhancements for dtn space communications," *International Journal of Satellite Communications and Networking*, vol. 34, no. 5, pp. 695–709, 2016.

[25] N. Bezirgiannidis, C. Caini, D. D. Padalino Montenero, M. Ruggieri, and V. Tsaoussidis, "Contact graph routing enhancements for delay tolerant space communications," in *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Sep. 2014, pp. 17–23.

[26] The Consultative Committee for Space Data Systems (CCSDS), *CCSDS Bundle Protocol Specification*, September 2015, CCSDS 734.2-B-1, Blue Book. [Online]. Available: https://public.ccsds.org/Pubs/734x2b1.pdf

[27] J. A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh and R. Velazco, "Assessing contact graph routing performance and reliability in distributed satellite constellations," *Journal of Computer Networks and Communications*, vol. 2017, 2017.

[28] A. Sekhar, B. S. Manoj, and C. S. R. Murthy, "MARVIN: movement-aware routing over interplanetary networks," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, Oct 2004, pp. 245–254.

[29] K. W. Leucht, "How nasa will use robots to create rocket fuel from Martian soil," *IEEE Spectrum*, 2018. [Online]. Available: https://spectrum.ieee.org/aerospace/robotic-exploration/how-nasa-will-use-robots-to-create-rocket-fuel-from-martian-soil

[30] K. Scott and S. Burleigh. (2007, November) Bundle Protocol (BP). IETF Request for Comments, RFC 5050. [Online]. Available: http://tools.ietf.org/html/rfc5050

[31] Y. Song, L. Liu, and J. Chen, "A temporal graph model based power aware routing algorithm in deep-space networks," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, May 2017, pp. 25–30.

[32] S. Burleigh, "Contact Graph Routing: draft-burleigh-dtnrg-cgr-01," Juy 2010. [Online]. Available: https://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-01

[33] J. Segui, E. Jennings, and S. Burleigh, "Enhancing contact graph routing for delay tolerant space networking," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Dec 2011, pp. 1–6.

[34] J. A. Fraire, P. Madoery, and J. M. Finochietto, "Leveraging routing performance and congestion avoidance in predictable delay tolerant networks," in *2014 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Oct 2014, pp. 1–7.

[35] S. Burleigh, C. Caini, J. J. Messina and M. Rodolfi, "Toward a unified routing framework for delay-tolerant networking," in *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Sept 2016, pp. 82–86.

[36] S. Dhara, S. Burleigh, R. Datta, and S. Ghose, "CGR-BF: An efficient contact utilization scheme for predictable deep space delay tolerant network," *Acta Astronautica*, vol. 151, pp. 401 – 411, 2018.

[37] The Consultative Committee for Space Data Systems (CCSDS), *Draft Recommended Standard for Schedule-Aware Bundle Routing*, July 2018, CCSDS 734.3-R-1, Red Book. [Online]. Available: https://public.ccsds.org/Lists/CCSDS%207343R1/734x3r1.pdf

[38] W. J. Wolfe and S. E. Sorensen, "Three scheduling algorithms applied to the Earth observing systems domain," *Management Science*, vol. 46, no. 1, pp. 148–166, 2000. [Online]. Available: http://www.jstor.org/stable/2634914

[39] Y. Xian, C. Huang, and J. Cobb, "Look-ahead routing and message scheduling in delay-tolerant networks," in *IEEE Local Computer Network Conference*, Oct 2010, pp. 40–47.

[40] K. Yoon and C. Hwang, *Multiple Attribute Decision Making: An Introduction*, ser. Multiple Attribute Decision Making: An Introduction. SAGE Publications, 1995, no. nos. 102-104.

[41] C.-L. Chang, *A modified VIKOR method for multiple criteria analysis*. London: Springer, 9 2010, vol. 168, no. 1-4, pp. 339–344.

[42] J. Brans, P. Vincke, and B. Mareschal, "How to select and how to rank projects: The promethee method," *European Journal of Operational Research*, vol. 24, no. 2, pp. 228 – 238, 1986, Mathematical Programming Multiple Criteria Decision Making.

[43] R. V. Rao, *Applications of Improved MADM Methods to the Decision Making Problems of Manufacturing Environment*. London: Springer, 8 2013, springer Series in Advanced Manufacturing.

[44] D. Sabaei, J. Erkoyuncu, and R. Roy, "A review of multi-criteria decision making methods for enhanced maintenance delivery," *Procedia CIRP*, vol. 37, pp. 30 – 35, 2015, cIRPe 2015 - Understanding the life cycle implications of manufacturing.

[45] T. P and B. V, "Experimental Evaluation of Suitability of Selected Multi-Criteria Decision-Making Methods for Large-Scale Agent-Based Simulations," *PLoS ONE*, vol. 11, no. 11, November 2016.

[46] I. Bisio, M. Marchese, and T. de Cola, "Congestion aware routing strategies for dtn-based interplanetary networks," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[47] IBM. Cplex optimmizer. [Online]. Available: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/#

[48] WebGeocalc Tool. [Online]. Available: http://naif.jpl.nasa.gov/naif/webgeocalc.html

[49] T. L. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, vol. 15, no. 3, pp. 234 – 281, 1977.

[50] T. Petr and B. Vladimír, "Experimental Evaluation of Suitability of Selected Multi-Criteria Decision-Making Methods for Large-Scale Agent-Based Simulations," *PLOS ONE*, vol. 11, no. 11, pp. 1–24, 11 2016.

[51] G.-H. Tzeng and J.-J. Huang, *Multiple Attribute Decision Making*. CRC Press, 2011.

[52] J. R. S. C. Mateo, *PROMETHEE*. London: Springer London, 2012, pp. 23–32.

[53] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[54] S. C. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in dtns," in *IEEE INFOCOM 2009*, April 2009, pp. 846–854.

[55] ION Code. [Online]. Available: http://ion-dtn.sourceforge.net/

[56] A. Berlati, S. Burleigh, C. Caini, F. Fiorini, J. J. Messina, S. Pozza, M. Rodolfi, and G. Tempesta, "Implementation of (o-)cgr in the one," in *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, Sept 2017, pp. 132–135.

[57] Systems Tool Kit. [Online]. Available: http://www.agi.com/products/stk/

[58] N. Alessi, C. Caini, T. de Cola, S. Martin, and J. P. Mayer, "DTN Performance in Complex Deep-Space Networks," in *2018 9th Advanced Satellite Multimedia Systems Conference and the 15th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Sep. 2018, pp. 1–7.

[59] J. Taylor, A. Makovsky, A. Barbieri, R. Tung, P. Estabrook, and A. G. Thomas, *Deep Space Communications*. Deep Space Communications and Navigation

Center of Excellence (JPL DESCANSO), October 2014, vol. 13, ch. 7, pp. 274–279.

[60] J. Taylor, D. K. Lee, and S. Shambayati, *Deep Space Communications*. Deep Space Communications and Navigation Center of Excellence (JPL DESCANSO, October 2014, vol. 13, ch. 6, p. 226.

[61] A. Makovsky, A. Barbieri, and R. Tung, *Odyssey Telecommunications*. Deep Space Communications and Navigation Center of Excellence (JPL DESCANSO), October 2002, ch. 5, pp. 37–56.

[62] European Space Agency (ESA). Mars Express Operations. [Online]. Available: http://www.esa.int/Our_Activities/Operations/Mars_Express_operations

[63] A. Gupta, A. Bansal, D. Naryani, and D. Kr. Sharma, "CRPO: Cognitive routing protocol for opportunistic networks," in *Proceedings of the International Conference on High Performance Compilation, Computing and Communications*, ser. HP3C-2017. New York, NY, USA: ACM, 2017, pp. 121–125.

[64] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, Na. Shahriar, F. Estrada-Solano, and O. M. Caicedo , "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[65] G. White, A. Palade, C. Cabrera, and S. Clarke, "Iotpredict: Collaborative qos prediction in iot," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2018, pp. 1–10.

[66] K. Nowicki and T. Uhl, *QoS/QoE in the Heterogeneous Internet of Things (IoT)*. Cham: Springer International Publishing, 2017, pp. 165–196.

[67] D. Mu, Y. Ge, M. Sha, S. Paul, N. Ravichandra and S. Chowdhury, "Adaptive radio and transmission power selection for internet of things," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, June 2017, pp. 1–10.

[68] F. A. Zdarsky and J. B. Schmitt, "Enhancing mobile qos based on movement contracts," in *Twelfth IEEE International Workshop on Quality of Service, 2004. IWQOS 2004.*, June 2004, pp. 3–9.

[69] R. Duan, X. Chen, and T. Xing, "A qos architecture for iot," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, Oct 2011, pp. 717–720.

[70] I. Awan, M. Younas, and W. Naveed, "Modelling qos in iot applications," in *2014 17th International Conference on Network-Based Information Systems*, Sep. 2014, pp. 99–105.

[71] S. Cabrero, R. García, X. G. García, and D. Melendi, "CRAWDAD dataset oviedo/asturies-er (v. 2016-08-08)," Downloaded from https://crawdad.org/oviedo/asturies-er/20160808, Aug. 2016.

[72] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*, 1st ed. Chichester: John Wiley and Sons, 2000.

[73] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 28th International Conference on Neural Informa-*

tion Processing Systems - Volume 1, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 802–810.

[74] K. Ashton. (2009) That 'Internet of Things' Thing. [Online]. Available: http://www.rfidjournal.com/article/print/4986

[75] S. Kraijak and P. Tuwanut, "A survey on iot architectures, protocols, applications, security, privacy, real-world implementation and future trends," in *11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015)*, Sep 2015, pp. 1–6.

[76] M. J. Beevi, "A fair survey on internet of things (iot)," in *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, Feb 2016, pp. 1–6.

[77] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the iot world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, Aug 2018.

[78] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "Iot architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*, Dec 2016, pp. 731–738.

[79] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[80] R. Li, X. Xiao, S. Ni, H. Zheng and S. Xia, "Byte segment neural network for network traffic classification," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, June 2018, pp. 1–10.

[81] T. Song, X. Shi, and X. Ma, "QoS aware dynamic power scaling algorithms for deploying energy efficient routers," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Oct 2014, pp. 235–236.

[82] C. H. Liu, Z. Chen, J. Tang, J. Xu and C. Piao, "Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[83] J. Xu, J. Wang, Q. Qi, H. Sun and B. He, "IARA: An Intelligent Application-Aware VNF for Network Resource Allocation with Deep Learning," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2018, pp. 1–3.

[84] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, Nov 2017.

[85] Federal Communications Commission (FCC), "2018 Broadband Deployment Report," February 2018. [Online]. Available: https://www.fcc.gov/reports-research/reports/broadband-progress-reports/2018-broadband-deployment-report

[86] Y. Gu, F. Ren, Y. Ji, and J. Li, "The evolution of sink mobility management in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 507–524, First quarter 2016.

[87] F. Chen and R. Li, "Sink node placement strategies for wireless sensor networks," *Wirel. Pers. Commun.*, vol. 68, no. 2, pp. 303–319, Jan. 2013. [Online]. Available: http://dx.doi.org.libproxy.unl.edu/10.1007/s11277-011-0453-x

[88] Y. Faheem, S. Boudjit, and K. Chen, "Data dissemination strategies in mobile sink wireless sensor networks: A survey," in *2009 2nd IFIP Wireless Days (WD)*, Dec 2009, pp. 1–6.

[89] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Distributed mobile sink routing for wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 877–897, Second 2014.

[90] N. Vlajic and D. Stevanovic, "Sink mobility in wireless sensor networks: A (mis)match between theory and practice," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, ser. IWCMC '09. New York, NY, USA: ACM, 2009, pp. 386–393. [Online]. Available: http://doi.acm.org.libproxy.unl.edu/10.1145/1582379.1582464

[91] Z. Vincze, R. Vida, and A. Vidács, "On the efficiency of local information-based sink deployment in heterogeneous environments," in *Proceedings of the 3rd International Conference on Wireless Internet*, ser. WICON '07. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 37:1–37:8. [Online]. Available: http://dl.acm.org.libproxy.unl.edu/citation.cfm?id=1460047.1460094

[92] S. Latambale and S. Sirsikar, "A survey of various sink mobility based techniques in wireless sensor network," in *Proceedings of the ACM Symposium*

on Women in Research 2016, ser. WIR '16. New York, NY, USA: ACM, 2016, pp. 45–50. [Online]. Available: http://doi.acm.org.libproxy.unl.edu/10.1145/2909067.2909075

[93] K. Shekanayaki, A. Chakure, and A. Jain, "A survey of journey of cloud and its future," in *2015 International Conference on Computing Communication Control and Automation*, Feb 2015, pp. 60–64.

[94] F. Fatemi Moghaddam, M. B. Rohani, M. Ahmadi, T. Khodadadi, and K. Madadipouya, "Cloud computing: Vision, architecture and characteristics," in *2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, Aug 2015, pp. 1–6.

[95] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2244–2281, third quarter 2016.

[96] N. Loutas, E. Kamateri, F. Bosi, and K. Tarabanis, "Cloud computing interoperability: The state of play," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, Nov 2011, pp. 752–757.

[97] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 369–392, First 2014.

[98] D. Kovachev and R. Klamma, "Beyond the client-server architectures: A survey of mobile cloud techniques," in *2012 1st IEEE International Conference on Communications in China Workshops (ICCC)*, Aug 2012, pp. 20–25.

[99] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 369–392, First 2014.

[100] R. K. Lomotey and R. Deters, "Architectural designs from mobile cloud computing to ubiquitous cloud computing - survey," in *2014 IEEE World Congress on Services*, June 2014, pp. 418–425.

[101] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, First quarter 2018.

[102] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, third quarter 2017.

[103] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[104] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, third quarter 2017.

[105] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[106] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthq uarter 2017.

[107] OpenFog consortium . [Online]. Available: https://www.openfogconsortium.org/

[108] Amazon Web Services (AWS). [Online]. Available: https://aws.amazon.com/

[109] Microsoft Azure. [Online]. Available: https://azure.microsoft.com/en-us/

[110] Google Cloud. [Online]. Available: https://cloud.google.com/

[111] Alibaba Cloud. [Online]. Available: https://www.alibabacloud.com/

[112] IBM Cloud. [Online]. Available: https://www.ibm.com/cloud/

[113] Oracle Cloud. [Online]. Available: https://cloud.oracle.com/home

[114] VMware Cloud. [Online]. Available: https://cloud.vmware.com/

[115] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[116] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[117] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[118] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[119] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[120] Y. Zhang, "Resource scheduling and delay analysis for workflow in wireless small cloud," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 675–687, 2018.

[121] X. Song and J. W. S. Liu, "Performance of multiversion concurrency control algorithms in maintaining temporal consistency," in *Proceedings., Fourteenth Annual International Computer Software and Applications Conference*, 1990, pp. 132–139.

[122] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2731–2735.

[123] Y. Sun and B. Cyr, "Sampling for data freshness optimization: Non-linear age functions," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 204–219, 2019.

[124] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Towards an effective age of information: Remote estimation of a markov source," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 367–372.

[125] G. Stamatakis, N. Pappas, and A. Traganitis, "Optimal policies for status update generation in an iot device with heterogeneous traffic," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5315–5328, 2020.

[126] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, 2019.

[127] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2017, pp. 486–493.

[128] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Optimizing data freshness, throughput, and delay in multi-server information-update systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 2569–2573.

[129] A. Arafa and S. Ulukus, "Age minimization in energy harvesting communications: Energy-controlled delays," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1801–1805.

[130] V. Marbukh, "Towards managing age of network state information in challenged networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 1–2.

[131] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, 2016.

[132] N. Lu, B. Li, R. Srikant, and L. Ying, "Optimal distributed scheduling of real-time traffic with hard deadlines," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 4408–4413.

[133] N. Lu, B. Ji, and B. Li, "Age-based scheduling: Improving data freshness for wireless real-time traffic," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 191–200.

[134] D. Nadig, S. El Alaoui, and B. Ramamurthy, "ERGO: A Scalable Edge Computing Architecture for Ag-IoT," Poster presented at 3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20), 2020.

[135] E. A. Brewer, "Kubernetes and the Path to Cloud Native," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 167.

[136] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. USA: Princeton University Press, 2009.

[137] W. A. Chaovalitwongse, I. P. Androulakis, and P. M. Pardalos, *Quadratic integer programming: complexity and equivalent formsQuadratic Integer Programming: Complexity and Equivalent Forms*. Boston, MA: Springer US, 2009, pp. 3153–3159.